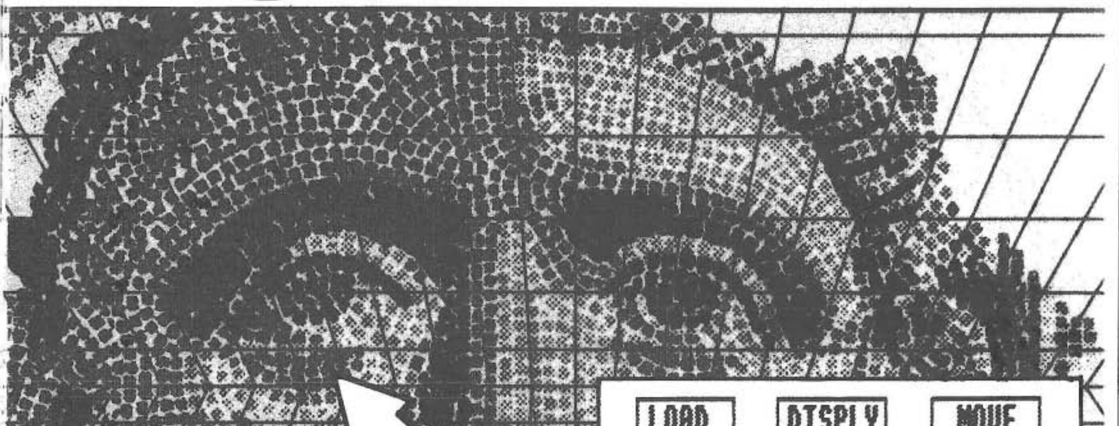
Volume 1
Issue 5

April
1988

Price £1.20

RISC USER



Screen Manager

THE MAGAZINE AND SUPPORT GROUP EXCLUSIVELY FOR USERS OF THE ARCHIMEDES

lume 1 Issue 5
ril 1988

RISC USER

CONTENTS

RISC User is published by BEEBUG Ltd.
Co-Editors: Mike Williams, Dr Lee Calcraft
Assistant Editor: Kristina Lucas
Production Assistant: Yolanda Turuelo
Technical Assistant: Lance Allison
Subscriptions: Mandy Mileham

BEEBUG, Dolphin Place, Holywell Hill, St.Albans, Herts AL1 1EX. Tel. St.Albans (0727) 40303

Printed by Arion House

The Archimedes Magazine
and Support Group.

EDITORIAL

CONTRIBUTING TO RISC USER

Like any magazine, we are always interested in publishing articles (including programs) ~~from~~ our readers. By necessity, many of the items in the first few issues have been written in house or commissioned from known authors. Even so, we have been very pleased to see that nearly every issue has contained one or more programs sent in to us by RISC User members. We would like to see many more programs and articles in the magazine originating in this way - after all, the Archimedes is such a rich machine that we cannot hope to delve into every aspect ourselves. And the more people there are who contribute their ideas and knowledge to provide articles for RISC User, the better the magazine will be for all Archimedes users.

However, it is easy to feel daunted, looking at some of the items we have published. But we can help. If you have an idea or embryo program which might be of interest to others, write to us with details. Equally, short technical articles, hints and tips are just as imporatant in producing an interesting and well balanced magazine. We can give you guidance on programming if required. We can also write the text to go with your program if necessary, or help you write your own. All of these things editorially we can provide, but we need your ideas and your knowledge of the Archimedes to make a start. And don't forget that we do pay for all contributions published in RISC User at rates of up to £40 per page.

To help would-be conributors we have prepared a short document, *Notes of Guidance for Contributors*, which is available on recept of an A5 size SAE. Share your knowledge of the Archimedes with other RISC User members so that we can all contribute to making the Archimedes the success it deserves to be.

RISC USER MAGAZINE DISC

We make no apolgy for mentioning the magazine disc within the editorial once again. Not only does this contain all the programs from this month's issue of RISC User, but once again we have managed to include some exciting additional items on the disc; a superb percussion sound library complete with demonstration program, and the first half of a really staggering graphics animation from Acorn. Demonstrations like these serve to show what can be achieved on an Archimedes. Further details of all the programs and other files on this month's disc are given on page 31.

NewsNewsNewsNewsNew

All the latest news and comment in the Archimedes world compiled by Mike Williams

0% FINANCE FROM BEEBUG

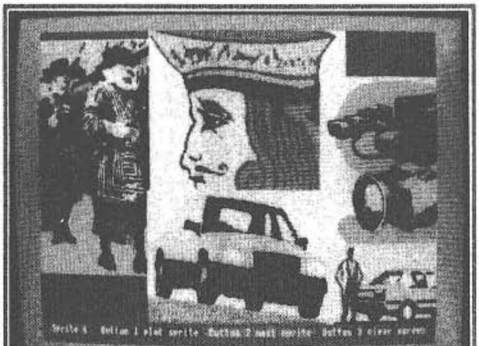
Although the 0% finance scheme provided by Acorn to help would-be users purchase an Archimedes has now lapsed, BEEBUG will continue to provide this same facility. This allows payment to be spread over 12 months at no extra cost. For more information contact BEEBUG on (0727) 40303.

ACORN NEWS

1st Word Plus, Acorn's major wordprocessor for the Archimedes, should be available in early April. We understand that the price has been fixed at £89.95 in. VAT, a little higher than we had anticipated (see review in RISC User Issue 4). Contact Acorn on (0223) 214411 for more information.

GRAPHICS STUDIO

New for the Archimedes is a set of three discs containing a graphics library of over 100 pictures. Each disc is divided into four sprite files from which sprites may be selected and displayed on-screen as required. The sprites may also be used in conjunction with Clares' Artisan, as well as in your own programs. Sample discs contained a variety of pictures including faces, birds, animals, cars etc. The *Archimedes Graphics Library* costs £21.5 from Micro Studio, 83 Clay Street, Soham, Cambridgeshire CB7 5HL or telephone (0353) 721736.


Sprite 4 Button 1 plot sprite Button 2 next sprite Button 3 clear screen

Pictures from the Archimedes Graphics Library

RESOURCES FOR THE ARCHIMEDES

Two further releases from Resource have come our way. *Droom* and *Desk Top Stories* are implementations of two very successful educational packages for the Archimedes. Both are aimed at primary school children, for use at school or at home. In either case the superbly enticing graphics are likely to delight both young and old. Droom provides a variety of problem solving exercises in the context of an adventure game, while Desk Top Stories enables young children to create illustrated stories. Both packs cost £18.95 direct from Resource, Exeter Road, Doncaster DN2 4PY or phone (0302) 63800.

MISCELLANY

C.J.E.Micro's has announced three products or those converting from a BBC micro to an Archimedes. Foremost among these is a fully buffered *Disc Drive Interface* for an external 5.25" drive costing £30.00 inc. VAT. A *Drive Swap Switch* may also be fitted for a further £8.00 allowing the external drive to appear as drive 0 (or drive A under the PC emulator). In addition C.J.E. have a *Drive Select Translation Cable* for £9.00 (this allows a drive configured internally as drive 0 to be reconfigured as drive 1 without having to change any internal links), and a utility called *Archie DFS Reader* (cost £15.00) which will transfer all your files from an external 5.25" DFS format disc to an Archimedes 3.5" disc. C.J.E.Micro's are on (0903) 213361.

GEM Electronics is a new company whose first product, *Desktop Games*, provides three logic games to be added to the other functions of Acorn's Desktop. The disc costs just £5.95 inc VAT and p&p direct from GEM Electronics, 17 Tandragee Road, Portadown, Craigavon BT62 3BQ.

Datanest Archimedes User Group has been launched in the Leicester area and will be meeting fortnightly. Contact Dick Brimson on leicester (0533) 741993 (work) or 873290 (home).

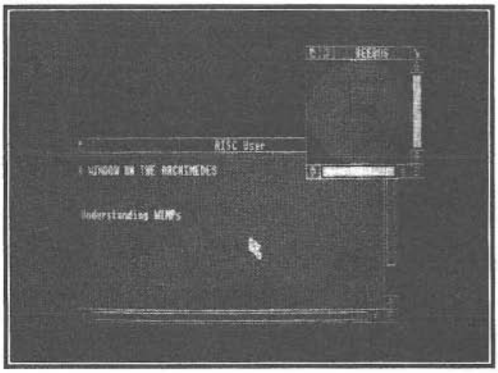
H.S.Software has converted 8 reading games from the BBC for the Archimedes. Aimed at 5-11 year olds, *Bumper Pack 1* costs £19.95 inc VAT direct from H.S. at 56 Hendrefoilan Avenue, Sketty, Swansea, West Glamorgan SA2 7NB or telephone (0792) 204519.

A smart *Mouse Mat* for your Archimedes is now available from BEEBUG for just £3.50. The mat comes in a choice of blue or red, with the BEEBUG logo top left, and the Archimedes logo bottom right. For more details see BEEBUG's Archimedes retail catalogue enclosed with this magazine. RU

A WINDOW ON THE ARCHIMEDES

One of the major new features on the Archimedes is a fully supported WIMP environment. Felix Andrew provides a down-to-earth introduction to this subject for Basic programmers.

Like many other computers, the Archimedes provides a comprehensive WIMP environment, that is a screen system using windows, icons, mouse and a pointer. All the functions to control the WIMP environment on the Archimedes are parcelled up in a module called WindowManager, but using these functions from within Basic is far from easy to understand on a first reading.



The purpose of this series of articles is to take the mystique out of using windows, and explain in simple steps how you can make the most of the features provided when writing your own programs. In so doing, I hope I will also be able to demonstrate the very real power of the WIMP Manager. If you are not too familiar with the general characteristics of WIMPs then I suggest you make a start by looking in the Welcome Guide on pages 20-25 which contains a quite useful introduction. Overall, there are three main features controlled by the WIMP Manager - windows, icons and menus.

WINDOWS

These are made up of a visible work area, which contains your graphics or text, surrounded by a window frame that is controlled by the WIMP manager. This includes the title bar, scroll boxes and so on. The visible work area is like a window onto a larger region which Acorn call the *work area extent* (*wae* or *page* for short).

SPRITES (OR ICONS)

Sprites may be text only like filenames, pre-defined sprites like the file icons in the Desktop, or editable text where you can type in information. Using a variety of these sprites, placed on a page, you can make your programs very much more user friendly.

MENUS

The menus on the Archimedes can *pop up* anywhere on the screen to display a list of options, and are very powerful yet quite simple to use. Each option on a menu can be made temporarily *non-selectable*, or may have a *tick* placed next to it to show the current (or default) status of that option.

THE PROGRAM

As a first step, we will deal thismonth with simple windows, and leave menus and icons for later. Unfortunately, using windows incurs a considerable overhead of code, even in the simplest of examples. The program listed this month is intended to provide a basis for future articles as well as demonstrating some simple windows as a starter. Begin by carefully typing in the program, keeping to the line numbering as given, and save it to disc before experimenting further.

When you run the program, you should see two windows appear on the screen. Play around with these windows, overlapping them, sending one behind the other, re-sizing them, just to see what effects can be achieved. All the window controls will operate correctly. Use Escape when you want to exit from the program.

The basic elements of the program are a procedure to create (define) windows (in this case two) called **PROCsetupwindows,** a procedure to open (display) each window called **PROCopenwinow1,** and a procedure to check for user input (from the mouse), **PROCpoll.** This latter is a key element in any WIMP program and controls the way in which windows and their contents are changed and manipulated on the screen.

The procedures call in turn the appropriate WIMP routines as defined in the Part 2 of the *Programmer's Reference Manual.* It is also essential that any WIMP program initialises the WIMP manager at the start, by a call to the routine **Wimp_Initialise**, and a corresponding call to **Wimp_CloseDown** on exit. You will be able to find all these features in this month's listing.

This is all we have space for this month. In the next issue we will examine in detail the use of the various procedures and functions involved so that you may use them as the basis of your own WIMP programs. You may well like to start experimenting straight away, but do be prepared for things not to work correctly at first. Perseverance will usually pay off in the end.

```
   10 REM > Wimps1
   20 REM Program   Window  Manager
   30 REM Author    Felix Andrew
   40 REM Version   A 1.0
   50 REM RISC User April 1988
   60 REM Program   Subject to Copyright
   70 :
   80 MODE12
   90 ON ERROR PROCerror:END
  100 PROCsetupvars:PROCsetupwindows
  110 PROCopenwindow1(w1,100,400,560,150
)
  120 PROCopenwindow1(w2,800,900,270,100
)
  130 REPEAT:PROCpoll:UNTIL quit=TRUE
  140 PROCshutdown
  150 END
  160 :
  170 DEF PROCsetupwindows
  180 w1=FNcreate(900,-600,c1,c0,c2,"RIS
C User",c1,c2,c0,c2,windowf)
  190 w2=FNcreate(300,-300,c1,c0,c3,"BEE
BUG",c1,c3,c0,c2,windowf)
  270 ENDPROC
  280 :
  290 DEF PROCshutdown
  300 PROCshut(w1):PROCshut(w2)
  310 PROCpoll
  320 SYS "Wimp_CloseDown"
  330 VDU 7:MODE12
  340 ENDPROC
  350 :
  360 DEF PROCerror
  370 ON ERROR OFF
  380 SYS"Wimp_CloseDown"
  390 VDU 7:MODE 12
  400 REPORT:PRINT" at line ";ERL
  410 ENDPROC
  420 :
  430 DEF PROCsetupvars
  440 VDU 19,0,16,192,96,112
  450 VDU 19,1,16,240,240,176
  460 VDU 19,2,16,48,8,240
  470 VDU 19,3,16,128,240,240
  475 VDU 19,15,16,64,144,160
  480 c0=0:c1=1:c2=2:c3=3:c15=15:GCOL c1
5+128:CLG
  490 DIM block% 512
  500 SYS "Wimp_Initialise" TO version
  510 mx=0:quit=FALSE
  570 REM Menu title flags
  580 title=1:move=2:vert=4:horiz=8
  590 nograph=16:noclose=128
  600 REM 'Macro title' flags
  610 alertf=nograph+noclose
  620 windowf=title+move+horiz+vert
  630 pulldownf=nograph+noclose
  660 string=-1:sprite=-2
  670 ENDPROC
  680 :
  690 DEF PROCpoll
  700 SYS "Wimp_Poll",0,block% TO flag,o
ffset
  720 CASE flag OF
  730  WHEN 1:PROCredrawwindow
  740  WHEN 2:PROCopenwindow2
  750  WHEN 3:PROCclosewindow
  790 ENDCASE
  800 ENDPROC
  810 :
  820 DEF PROCopenwindow2
  830 SYS "Wimp_OpenWindow",, offset
  840 ENDPROC
  850 :
  860 DEF PROCclosewindow
  870 handle=offset!0
  880 SYS "Wimp_CloseWindow",,offset
  890 ENDPROC
  900 :
  910 DEF PROCshut(handle)
  920 offset!0=handle
  930 SYS "Wimp_CloseWindow",,offset
  940 ENDPROC
 1140 :
 1150 DEF PROCredrawwindow
 1160 LOCAL flag,window
 1170 window=offset!0
 1180 block%!0=window
 1190 SYS "Wimp_GetWindowInfo",,block%
 1200 textcol=block%?38
 1210 SYS "Wimp_RedrawWindow",,offset TO
 flag
```

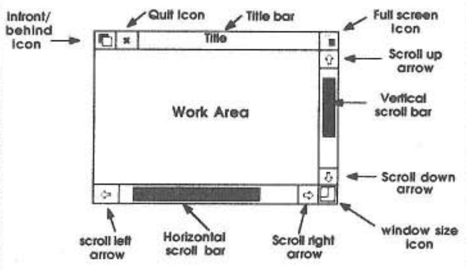


Figure 1. Archimedes Window Layout

```
 1220 WHILE flag<>0
 1230  PROCnewvals
 1240  SYS "Wimp_GetRectangle",,offset T
O flag
 1250  CASE window OF
 1260  WHEN w1:PROCwindow1
 1270  WHEN w2:PROCwindow2
 1280  ENDCASE
 1290 ENDWHILE
 1300 ENDPROC
 1310 :
 1320 DEF PROCtext(X,Y,T$)
 1330 IF FNin(X,Y-32,X+16*LEN T$,Y) THEN
 1340  PROCgwindow
 1350  MOVE X,Y
 1360  GCOL 0,textcol:PRINTT$:VDU26
 1370 ENDIF
 1380 ENDPROC
 1390 :
 1400 DEF FNin(A%,B%,C%,D%)
 1410 IF C%<ebx THEN =FALSE
 1420 IF A%>etx THEN =FALSE
 1430 IF B%>ety THEN =FALSE
 1440 IF D%<eby THEN =FALSE
 1450 =TRUE
 1460 :
 1470 DEF PROCnewvals
 1480 gbx=offset!28:gby=offset!32
 1490 gtx=offset!36:gty=offset!40
 1500 wbx=offset!4:wby=offset!8
 1510 wtx=offset!12:wty=offset!16
 1520 xsc=offset!20:ysc=offset!24
 1530 hp=wtx-wbx:vp=wty-wby:ebx=xsc
 1540 etx=xsc+hp:ety=ysc:eby=ysc-vp
 1550 ENDPROC
 1560 :
 1570 DEF PROCgwindow
 1580 VDU 26,5,24,gbx;gby;gtx-2;gty-4;
 1590 ORIGIN wbx-ebx,wty-ety
 1600 ENDPROC
 1610 :
 1620 DEF FNcreate(pw,pd,tf,tb,ts,T$,pf,
pb,sf,sb,tflags)
 1630 $block%=STRING$(84,CHR$0)
 1640 block%!24=-1:block%!28=tflags
 1650 block%?32=tf:block%?33=tb
 1660 block%?34=pf:block%?35=pb
 1670 block%?36=sf:block%?37=sb
 1680 block%?38=ts
 1690 block%?39=0:block%!40=0
 1700 block%!44=pd:block%!48=pw
 1710 block%!52=0:block%!56=&3D
 1720 block%!60=-1 AND &3000
 1730 block%!64=0:block%!68=0
 1740 T$=LEFT$(T$,11):$(block%+72)=T$
 1750 block%!84=0:block%!88=0
 1760 SYS "Wimp_CreateWindow",,block% TO
 handle
 1770 =handle
 2130 :
 2140 DEF PROCopenwindow1(handle,x,y,w,h
)
 2150 block%!0=handle
 2160 block%!4=x:block%!8=y-h
 2170 block%!12=x+w:block%!16=y
 2180 block%!20=0:block%!24=0
 2190 block%!28=-1
 2200 SYS "Wimp_OpenWindow",,block%
 2210 ENDPROC
 2220 :
 5000 DEF PROCwindow1
 5010 PROCtext(64,-64,"A WINDOW ON THE A
RCHIMEDES")
 5020 PROCtext(64,-168,"Understanding WI
MPs")
 5030 ENDPROC
 5040 :
 5050 DEF PROCwindow2
 5060 IF FNin(0,-300,300,0) THEN
 5070  PROCgwindow
 5080  GCOL c2
 5090  CIRCLE FILL 150,-150,150
 5100  VDU26
 5110 ENDIF
 5120 PROCtext(112,-50,"Acorn")
 5130 PROCtext(72,-130,"Archimedes")
 5150 ENDPROC
```

3D GRAPHICS

Mark Davis begins a short series on 3D graphics with a simple, but extremely fast wire frame manipulator in Basic.

The speed of the Archimedes, and its excellent graphics facilities makes it an ideal machine for experimenting with three-dimensional graphics. You will already have seen just what can be achieved from David Braben's excellent creations, Lander and Zarch. In this short series of articles, I will be giving a number of programs for handling three-dimensional graphics, progressing to hidden line removal and shading. This month we begin at the very beginning with a look at simple wire frame objects manipulated in Basic.

WIRE FRAME GRAPHICS

The simplest form of 3D graphics is the wire frame type, where a model is defined as a set of vertices and a list of connective information. This is the form of 3D graphics employed in this article. There are three basic steps to displaying 3D models. The first is to have an easy-to-use data entry system for vertices and connecting lines. The second is to be able to rotate the shape upon all axes, so it can be viewed from any angle. The final step is to convert the 3D co-ordinates into two-dimensional ones, so that the shape can be displayed on the screen.

The data entry system for the moment can simply consist of two sets of DATA statements, the first containing a list of 3D coordinates, and the second a list of connecting lines. The rotation can be achieved by simple trigonometry, and is extremely fast in Basic V, as is the perspective transformation.

THE PROGRAM

Carefully type in the listing, and save the program to disc. Run it, and you should be presented with a wire frame cube. Use the mouse to rotate it along the X and Y axes, the left and middle mouse buttons to bring it nearer or further away, and cursor keys to translate it in the X and Y directions.

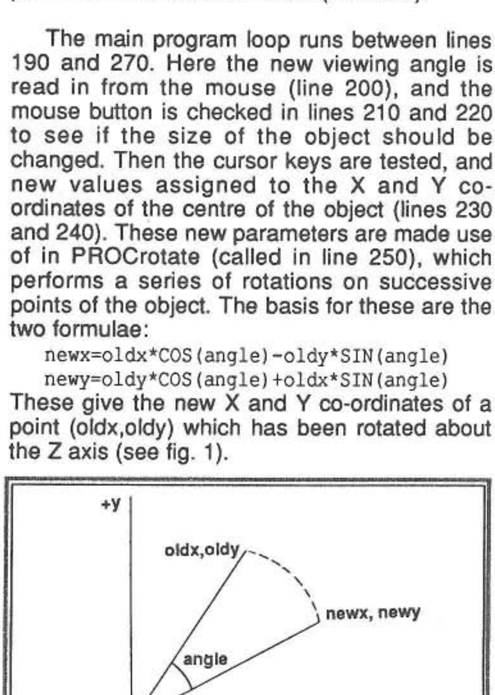
HOW IT WORKS

The operation of the program is very simple. Lines 80-110 set up a variety of arrays. Then lines 130 and 140 provide starting values for the position and angle of the object. Next PROCinit is called which reads in two sets of data. The first gives the co-ordinates of each point on the object, the second details of which points are connected to which (see later).

The main program loop runs between lines 190 and 270. Here the new viewing angle is read in from the mouse (line 200), and the mouse button is checked in lines 210 and 220 to see if the size of the object should be changed. Then the cursor keys are tested, and new values assigned to the X and Y co-ordinates of the centre of the object (lines 230 and 240). These new parameters are made use of in PROCrotate (called in line 250), which performs a series of rotations on successive points of the object. The basis for these are the two formulae:

```
newx=oldx*COS(angle)-oldy*SIN(angle)
newy=oldy*COS(angle)+oldx*SIN(angle)
```

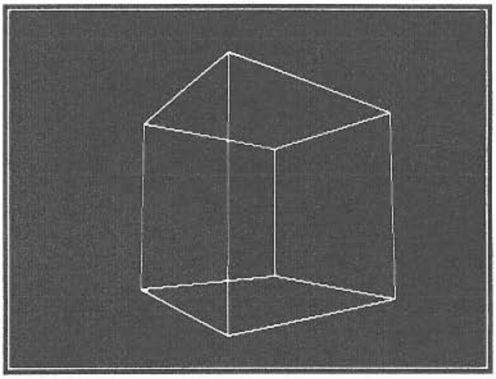
These give the new X and Y co-ordinates of a point (oldx,oldy) which has been rotated about the Z axis (see fig. 1).



```
newx = oldx*COS(angle) - oldy*SIN(angle)
newy = oldy*COS(angle) + oldx*SIN(angle)
```

Figure 1 - Rotation upon the Z axis

Once the rotations have been calculated, the transformation of the 3D object to its 2D representation is accomplished in lines 620 and 630. The two factors of 800 determine the degree of perspective in the Z direction, and at the same time the on-screen size of the object. The smaller the factor, the greater the perspective. Try the program with the value 200. This gives the kind of perspective which you would get if the object was the size of a house, rather than a shoe box.



CREATING YOUR OWN SHAPE

Creating your own shape is easy enough, especially if you use a diagram. First of all you must work out how many different points are needed to define the shape (e.g. 8 for a cube), and change the value of *coords* in line 310. Then type the co-ordinates in as DATA statements replacing lines 390-420.

xw(),yw(),zw()	data for the 3D shape
xr(),yr(),zr()	real co-ordinates after rotation and translation
xs(),ys()	screen co-ordinates
plot()	plot types for link data
point()	co-ord ref nos for link data
xrot,yrot,zrot	rotation values for each axis
xpos,ypos,zpos	3D origin for shape
coords	number of co-ordinates
lines	total amount of link data

Fig 2. Variables used.

Next you must find the most economical way of joining the points up to make the complete 3D shape. The data for the lines (or links) is encoded in pairs of numbers (lines 450-470 in the program. The first of each pair must be the number 4, indicating a MOVE operation, or a 5 indicating a DRAW. The second number is the number of the apex referred to. They are numbered from zero to 7 in the case of the cube, and correspond to the order in which the point data was given in the succeeding DATA lines. For example the first six numbers on line 450: 4, 0, 5, 1, 5, 2 mean "move to point zero (i.e. 10,-10,-10), then draw a line to point number 1, and a line from point 1 to point 2", and so on.

Once this is complete, you should change the value of the variable *lines* in line 310 to reflect the number of pairs of connection data supplied. Finally you should set the values in the DIM statements from lines 80-100 to equal or exceed the total number of points (the variable *coords*), and set the values in line 110 to equal or exceed the total number of link values (the variable *lines* in line 310).

The program is now ready to run, and you may view your shape from any angle. You may find that with a large number of points, the object will flicker, and may be slow to draw. The ARM assembler program to be published next month should solve this problem, because it will work at a considerably higher speed, and with quite complex objects.

```
   10 REM >BasicWire
   20 REM Program   3D object drawing
   30 REM Version   A 1.03
   40 REM Author    Mark J. Davis
   50 REM RISC User April 1988
   60 REM Program   Subject to Copyright
   70 :
   80 DIM xw(20),yw(20),zw(20)
   90 DIM xr(20),yr(20),zr(20)
  100 DIM xs(20),ys(20)
  110 DIM plot(20),point(20)
  120 MODE0:ORIGIN 640,512:OFF
  130 xrot=0:yrot=0:zrot=0
  140 xpos=0:ypos=0:zpos=100
  150 PROCinit
  160 *POINTER
  170 MOUSE ON 0:MOUSE TO 0,0
  180 MOUSE RECTANGLE -1000,-1000,2000,2
000
  190 REPEAT
  200 MOUSE yrot,xrot,B
  210 IF B AND 4 THEN zpos+=(zpos>20)*4
  220 IF B AND 2 THEN zpos+=4
  230 xpos+=(INKEY(-26)-INKEY(-122))*2
  240 ypos+=(INKEY(-42)-INKEY(-58))*2
  250 PROCrotate
```

```
260 WAIT:CLS:PROCplot
270 UNTIL FALSE
280 END
290 :
300 DEFPROCinit
310 coords=8:lines=16:RESTORE 390
320 FOR X=0 TO coords-1
330 READ xw(X),yw(X),zw(X):NEXT
340 FOR X=1 TO lines
350 READ plot(X),point(X):NEXT
360 ENDPROC
370 :
380 REM Co-ordinate data
390 DATA 10,-10,-10, 10,10,-10
400 DATA 10,10,10, 10,-10,10
410 DATA -10,-10,-10, -10,10,-10
420 DATA -10,10,10, -10,-10,10
430 :
440 REM link data
450 DATA 4,0, 5,1, 5,2, 5,3, 5,0, 5,4
460 DATA 5,7, 5,3, 4,2, 5,6, 5,5, 5,1
470 DATA 4,4, 5,5, 4,6, 5,7
480 :
490 DEFPROCrotate
500 FOR I%=0 TO coords-1
510 xa=xw(I%):ya=yw(I%):za=zw(I%)
520 xr=xa*COS(RAD(zrot))-ya*SIN(RAD(zrot))
530 ya=ya*COS(RAD(zrot))+xa*SIN(RAD(zrot))
540 xa=xr
550 xr=xa*COS(RAD(yrot))-za*SIN(RAD(yrot))
560 za=za*COS(RAD(yrot))+xa*SIN(RAD(yrot))
570 xa=xr
580 yr=ya*COS(RAD(xrot))-za*SIN(RAD(xrot))
590 za=za*COS(RAD(xrot))+ya*SIN(RAD(xrot))
600 ya=yr
610 xr(I%)=xa+xpos:yr(I%)=ya+ypos:zr(I%)=za+zpos
620 xs(I%)=800*xr(I%)/zr(I%)
630 ys(I%)=800*yr(I%)/zr(I%)
640 NEXT:ENDPROC
650 :
660 DEFPROCplot
670 FORP%=1TOlines
680 PLOT plot(P%),xs(point(P%)),ys(point(P%))
690 NEXT:ENDPROC
```

ARCHIMEDES SPEED WARNING

Although it is a well known fact that the Archimedes is fast, it is less well known that its speed depends quite dramatically on the screen mode in use. The accompanying table reveals all. It shows the speed in each mode for three different tasks. In each case the speed is given as a ratio of the speed taken in mode 0. The three tasks are:

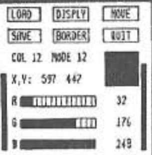
1. An empty FOR loop to 10000
2. Draw 100 large crosses
3. Draw 100 large filled rectangles

Even with the first test, mode 12 turned in a result some 14% slower than mode 0, while modes 15 and 20 were a massive 40% slower than mode 0. On the graphics-intensive third test, the high resolution modes did particularly badly. Mode 12 was 78% slower than mode 0, with modes 15 and 20 trailing at 197% and 330% respectively.

So the moral is simple: make your choice of mode with great care. And if there are time critical elements to a program, perform them while in the "fastest" suitable mode.

Lee Calcraft

MODE	SIZE	TEST1	TEST2	TEST3
0	20K	1.00	1.00	1.00
1	20K	1.00	0.72	1.00
2	40K	0.97	0.70	1.37
3	40K	1.03	....	....
4	20K	1.03	0.73	1.00
5	20K	1.00	0.65	1.01
6	20K	1.00	....	....
7	80K	1.03	....	....
8	40K	1.03	1.04	1.36
9	40K	1.05	0.76	1.37
10	80K	1.14	0.79	1.38
11	40K	1.05	....	....
12	80K	1.14	1.18	1.78
13	80K	1.14	0.87	1.77
14	80K	1.14	....	....
15	160K	1.43	1.54	2.97
16	132K	1.30	....	....
17	132K	1.30	....	....
18	40K	1.03	1.38	2.03
19	80K	1.11	1.54	2.92
20	160K	1.41	1.94	4.30



THE RISC USER SCREEN MANAGER

by Lee Calcraft

Use this program to help you with the colour and layout of your screen designs.

The Screen Manager described here provides a pop-up window on any 80 column Archimedes screen to give the user full control of the colours used in the non-256 colour modes, together with a running display of the co-ordinates of the pointer. This latter feature can be very useful when designing a display screen of one kind or another, since it allows you to move the pointer to a particular position, and then read off the corresponding co-ordinates for later use in PLOT and DRAW commands.

But the principal function of the Screen Manager (listing 1) is to give full control over the palette in non-256 colour modes. The Manager is supplied as a procedure which can be appended to a program under development, or just called from immediate mode Basic with:

```
PROCscreenman
```

The result of this call is to display a window similar to that in the accompanying illustration.

As the pointer is moved around the screen, the display is continually updated to feature the colour of the pixel at the pointer. Its colour number, and red, green and blue components are also given. The latter appear both numerically, and in the form of a horizontal bar display. The current colour is also displayed as a small coloured square within the main window. If the *select* button is pressed, the program brings the colour faders into operation, when the *menu* (middle) button of the mouse can be used to alter the composition of the current colour by moving the pointer across the corresponding bar display. To *fix* a colour so created, press the *adjust* button. Alternatively, pressing *select* will return the colour to its original value.

In this way it is possible to experiment with combinations of colours for a particular application. For example, it is only by using such a device that you can create the best shading effects, and can determine which colours complement each other and which do not. Once you have created a set of colours which suit your purpose, you may save them away to disc as a special palette file by using the *select* button over the "SAVE" box at the top of the window. Similarly the "LOAD" option will let you load in a previously saved palette file. Incidentally, star commands can be issued at the filename prompt with either of these options.

The other four boxes at the top of the window work as follows. "BORDER" allows you to adjust the colour of the screen border (which is also saved in the palette file), "DISPLY" displays the colour definitions of the colours of the current palette. "QUIT" takes you out of the Screen Manager, and returns you to the program from which it was called. Pressing Return at any time has the same effect, and you will notice that in either case the area of screen obscured by the Screen Manager's window is reinstated on exit. Finally, the "MOVE" option allows you to move the window around the screen, so that it can be positioned at the most convenient place for any given task.

USING THE PROGRAM

You will see that there are two listings accompanying this article. The second simply sets up a coloured mode 12 screen with a variety of shading and 3D and engraved text, and then calls the Screen Manager. You may choose not to type in this listing, though you will probably find it a useful testbed for experimenting with colour combinations for other displays.

The Screen Manager proper appears in listing 1, and should be typed in and saved away. To make use of it, simply load in the program which creates the screen to be edited, and type:

```
APPEND "ScrnMan2"
```

The appending operation automatically renumbers the incoming program as necessary, so you can ignore line numbers. Next insert the call:

```
PROCscreenman
```

at the appropriate point in your own program, and the window will appear when this line is reached. If you are using listing 1 then you do not need to insert the procedure call, since it is already in place.

PROGRAM NOTES

The Screen Manager is best suited to the very popular mode 12 and to the high resolution mode 20 (multi-sync monitor permitting), but it will work quite satisfactorily in modes 0 and 8. In these two modes however, each colour is displayed more than once (4 times in mode 8, and 8 times in mode 0). The program can also be used in mode 15, where the pointer co-ordinate display can be useful. In this mode the colour number display also functions correctly, but the colour re-definer using the slider bars does not. If there is sufficient interest I will add a routine to extend the facilities offered in mode 15.

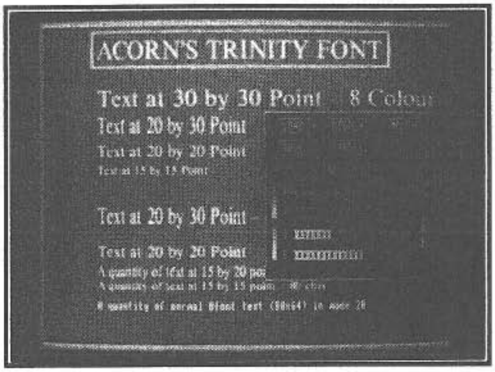
The program includes full error trapping using the special LOCAL ERROR call to handle errors within the Screen Manager procedure without exiting to the calling program.

Next month I will briefly show how to make use of palette files saved by the Screen Manager.

Spritesize: Mode12 - 24K, Mode 15, 20 - 40K

Listing 1

```
   10 REM            >ScrnMan2
   20 REM Program    Screen Manager
   30 REM Version    A 0.2
   40 REM Author     Lee Calcraft
   50 REM RISC User  April 1988
   60 REM Program    Subject to copyright
   70 :
   80 REM------SCREEN MANAGER-----------
   90 :
  100 DEFPROCscreenman
  110 LOCAL A,B,WX,WY,X,X1,X2,Y,Y1,Y2,Z
  120 LOCAL border,blue,blue$,dummy,firs
t
  130 LOCAL f,file$,green,green$,ired
  140 LOCAL igreen,iblue,logcol,move
  150 LOCAL oldpfile$,oldX,oldY,second
  160 LOCAL sred,sgreen,sblue,xbase
  170 LOCAL xchr,ybase,ychr,pos,quit,red
  180 PROCceinit
  190 LOCAL ERROR
  200 ON ERROR LOCAL PROClocalerr
  210 REPEAT
  220 IF move THEN PROCmove:move=FALSE
  230 PROCcewindow
  240 REPEAT
  250 PROCcepointer
  260 PROCcefader
  270 IF Z=4 THEN PROCbutton
  280 UNTIL quit OR move
  290 UNTIL quit
  300 PROCrestore
  310 RESTORE ERROR
  320 ENDPROC
  330 :
  340 DEFPROCmove
  350 *SCHOOSE origscrn
  360 PLOT &ED,WX,WY
  370 GCOL3,7
  380 REPEAT
  390 MOUSE X,Y,Z
  400 X=(400-X)*(X>400)
  410 Y=(460-Y)*(Y>460)
  420 RECTANGLE X,Y,500,500
  430 dummy=INKEY(5)
  440 RECTANGLE X,Y,500,500
  450 UNTIL Z=0
  460 PROCnewbase
  470 ENDPROC
  480 :
  490 DEFPROCnewbase
  500 ybase=32*(Y DIV 32)+4:IF ybase>516
 ybase=516
  510 xbase=16*(X DIV 16)+14:IF xbase>76
6 xbase=766
  520 xchr=(xbase+14)/16+1:ychr=(1024-yb
ase)/32-1
  530 VDU28,xchr,f*ychr,xchr+28,f*(ychr-
13)
  540 WX=xbase:WY=ybase
  550 MOVE WX,WY:MOVE WX+500,WY+500
  560 *SGET origscrn
  570 ENDPROC
  580 :
  590 DEFPROCbutton
  600 CASE TRUE OF
  610 WHEN FNselect(WX+39,WX+139,WY+441,
WY+481):PROCpalload
  620 WHEN FNselect(WX+39,WX+139,WY+379,
WY+419):PROCpalsave
  630 WHEN FNselect(WX+184,WX+296,WY+441
,WY+481):PROCpaldisp
  640 WHEN FNselect(WX+345,WX+457,WY+441
,WY+481):move=TRUE
  650 WHEN FNselect(WX+345,WX+457,WY+379
,WY+419):quit=TRUE
  660 WHEN FNselect(WX+184,WX+296,WY+379
,WY+419):PROCborder:PROCcefader:PROCcead
just
  670 OTHERWISE PROCceadjust
  680 ENDCASE
  690 ENDPROC
  700 :
  710 DEFPROCrestore
  720 *SCHOOSE origscrn
  730 PLOT &ED,WX,WY
  740 VDU26:PRINTTAB(0,f*31);:ON:*FX15
  750 *POINTER 0
  760 ENDPROC
  770 :
  780 DEFPROCceinit
  790 OFF:*POINTER
  800 f=1-(MODE=20):*FX9
  810 X=750:Y=100
```


ACORN'S TRINITY FONT

Text at 30 by 30 Point 8 Colou

Text at 20 by 30 Point

Text at 20 by 20 Point

Text at 15 by 15 Point

Text at 20 by 30 Point

Text at 20 by 20 Point

A quantity of text at 15 by 20 poi

A quantity of text at 15 by 15 point [illegible]

A quantity of normal 8font text (80x64) in mode 20

```
  820 PROCnewbase
  830 border=FALSE:quit=FALSE:move=FALSE
  840 oldpfile$="None Known"
  850 VDU23,255,255,255,63,63,63,63,255,
255
  860 VDU23,254,255,1,1,1,1,1,1,255
  870 VDU23,253,1,1,1,1,1,1,1,1
  880 COLOUR128:COLOUR7
  890 ENDPROC
  900 :
  910 DEFPROCclearwind
  920 GCOL 0:RECTANGLE FILL WX,WY,500,50
0
  930 GCOL 7
  940 RECTANGLE WX,WY,500,500
  950 ENDPROC
  960 :
  970 DEFPROCcewindow
  980 PROCclearwind
  990 PRINTTAB(10,f*4)"MODE ";MODE
 1000 PRINTTAB(2,0)"LOAD"SPC5"DISPLY"SPC
5"MOVE"
 1010 PRINTTAB(2,f*2)"SAVE"SPC5"BORDER"S
PC5"QUIT"
 1020 RECTANGLE WX+39,WY+379,100,40
 1030 RECTANGLE WX+184,WY+379,112,40
 1040 RECTANGLE WX+345,WY+379,112,40
 1050 RECTANGLE WX+39,WY+441,100,40
 1060 RECTANGLE WX+184,WY+441,112,40
 1070 RECTANGLE WX+345,WY+441,112,40
 1080 FOR A=0 TO 7
 1090 COLOUR 128+A
 1100 PRINT TAB(0,f*(A+5))" ";
 1110 NEXT
 1120 FOR A=8 TO 15
 1130 COLOUR 128+A
 1140 PRINTTAB(28,f*(A-3))" ";
 1150 NEXT
 1160 COLOUR 128
 1170 ENDPROC
 1180 :
 1190 DEFPROCcepointer
 1200 MOUSE X,Y,Z
 1210 logcol=POINT(X,Y)
 1220 PRINTTAB(2,f*4)"COL ";logcol;SPC(2
)
 1230 GCOL logcol:RECTANGLEFILL WX+350,W
Y+250,100,100
 1240 first=FNreadcolour(logcol)
 1250 PRINTTAB(2,f*6)"X,Y:  ";X TAB(11);
Y SPC3;
 1260 ENDPROC
 1270 :
 1280 DEFFNreadcolour(logcol)
 1290 SYS "OS_ReadPalette",logcol,16-8*b
order TO A,B,first,second
 1300 blue=first>>>24
 1310 green=(first>>>16)  AND &FF
 1320 red=(first>>>8) AND &FF
 1330 =first
 1340 :
 1350 DEFPROCcefader
 1360 ired=red>>>4:igreen=green>>>4:iblu
e=blue>>>4
 1370 red$="R"+CHR$253+STRING$(ired,CHR$
255)+STRING$(15-ired,CHR$254)
 1380 green$="G"+CHR$253+STRING$(igreen,
CHR$255)+STRING$(15-igreen,CHR$254)
 1390 blue$="B"+CHR$253+STRING$(iblue,CH
R$255)+STRING$(15-iblue,CHR$254)
 1400 PRINTTAB(2,f*8)red$SPC4;red;SPC(2)
 1410 PRINTTAB(2,f*10)green$SPC4;green;S
PC(2)
 1420 PRINTTAB(2,f*12)blue$SPC4;blue;SPC
(2)
 1430 IF INKEY(-74)THEN quit=TRUE
 1440 ENDPROC
 1450 :
 1460 DEFPROCceadjust
 1470 PRINTTAB(2,f*6)"RGB "TAB(5);red TA
B(9);green TAB(13);blue;"  "
 1480 oldX=X:oldY=Y
 1490 MOUSE RECTANGLE WX,WY,500,230
 1500 sred=red:sgreen=green:sblue=blue
 1510 REPEAT:MOUSE X,Y,Z:UNTIL Z=0
 1520 REPEAT
 1530 REPEAT
 1540 MOUSE X,Y,Z
 1550 UNTIL Z=1 OR Z=2 OR Z=4
 1560 IF Z=1 OR Z=4 THEN Y=0
 1570 IF Y>=WY+192 AND Y<=WY+220 THEN ir
ed=FNfadepos:red=ired<<4
 1580 IF Y>=WY+126 AND Y<=WY+155 THEN ig
reen=FNfadepos:green=igreen<<4
 1590 IF Y>=WY+63 AND Y<=WY+90 THEN iblu
```

```
e=FNfadepos:blue=iblue<<4
 1600 IF Y>0 THEN PROCcefader
 1610 IF border THEN
 1620 VDU19,0,24,red,green,blue
 1630 ELSE COLOUR logcol,red,green,blue
 1640 ENDIF
 1650 UNTIL Z=1 OR Z=4 OR quit=TRUE
 1660 MOUSE RECTANGLE 0,0,1279,1023
 1670 MOUSE TO oldX,oldY
 1680 IF Z=4 THEN red=sred:green=sgreen:
blue=sblue
 1690 IF border THEN
 1700 VDU19,0,24,red,green,blue
 1710 ELSE COLOUR logcol,red,green,blue
 1720 ENDIF
 1730 REPEAT:MOUSE X,Y,Z:UNTIL Z<>4
 1740 border=FALSE
 1750 ENDPROC
 1760 :
 1770 DEFFNfadepos
 1780 pos=(X-(WX+64))/16
 1790 IF pos<0 pos=0
 1800 IF pos>15 pos=15
 1810 =pos
 1820 :
 1830 DEFPROCpalload
 1840 REPEAT
 1850 PROCclearwind
 1860 ON:PRINTTAB(7,0)"Load Palette"'
 1870 PRINT"Previously "oldpfile$
 1880 INPUT"Filename   "file$
 1890 IF LEFT$(file$,1)="*" THEN PROCsta
r
 1900 UNTIL LEFT$(file$,1)<>"*"
 1910 PROCloader(file$)
 1920 CLS:OFF
 1930 PROCcewindow
 1940 ENDPROC
 1950 :
 1960 DEFPROCloader(file$)
 1970 oldpfile$=file$
 1980 CLOSE #0
 1990 D%=OPENIN(file$)
 2000 FOR COL=0 TO 15
 2010 PROCget
 2020 COLOUR COL,red,green,blue
 2030 NEXT
 2040 PROCget
 2050 VDU19,0,24,red,green,blue
 2060 CLOSE #D%
 2070 ENDPROC
 2080 :
 2090 DEFPROCget
 2100 red=BGET#D%
 2110 green=BGET#D%
 2120 blue=BGET#D%
 2130 ENDPROC
 2140 :
 2150 DEFPROCpalsave
 2160 REPEAT
 2170 PROCclearwind
 2180 ON:PRINTTAB(7,0)"Save Palette"'
 2190 PRINT"Previously "oldpfile$
 2200 INPUT"Filename   "file$
 2210 IF LEFT$(file$,1)="*" THEN PROCsta
r
 2220 UNTIL LEFT$(file$,1)<>"*"
 2230 oldpfile$=file$
 2240 CLOSE #0
 2250 D%=OPENOUT(file$)
 2260 FOR COL=0 TO 15
 2270 PROCreadcolour(COL)
 2280 PROCput
 2290 NEXT
 2300 border=TRUE
 2310 first=FNreadcolour(0)
 2320 border=FALSE
 2330 PROCput
 2340 CLOSE #D%:CLS:OFF
 2350 PROCcewindow
 2360 ENDPROC
 2370 :
 2380 DEFPROCput
 2390 BPUT#D%,red
 2400 BPUT#D%,green
 2410 BPUT#D%,blue
 2420 ENDPROC
 2430 :
 2440 DEFPROCreadcolour(logcol)
 2450 SYS "OS_ReadPalette",logcol,16 TO
A,B,first,second
 2460 blue=first>>>24
 2470 green=(first>>>16)  AND &FF
 2480 red=(first>>>8) AND &FF
 2490 ENDPROC
 2500 :
 2510 DEFPROCstar
 2520 CLS:VDU14:OSCLI(file$)
 2530 PRINT'"Press space ";
 2540 IF GET CLS:VDU15
 2550 ENDPROC
 2560 :
 2570 DEFPROCborder
 2580 border=TRUE
 2590 first=FNreadcolour(90)
 2600 ENDPROC
 2610 :
 2620 DEFPROClocalerr
 2630 OSCLI("FX4"):CLOSE#0:PROCclearwind
:VDU30:REPORT
 2640 IF (ERR AND &FF) <128 THEN PRINT"
at line ";ERL ELSE PRINT
```

```
 2650 MOUSE RECTANGLE 0,0,1279,1023
 2660 IF ERR=17 PROCerrget ELSE PROCspac
eget
 2670 OFF:ENDPROC
 2680 :
 2690 DEFPROCspaceget
 2700 PRINT'"Press space ";
 2710 REPEAT UNTIL GET=32
 2720 ENDPROC
 2730 :
 2740 DEFPROCerrget
 2750 PRINT'"Press space to continue"
 2760 PRINT"or 'Q' to quit ";
 2770 REPEAT:A=GET:UNTIL A=32 OR A=81 OR
 A=113
 2780 IF A>32 THEN MODE 12:END
 2790 ENDPROC
 2800 :
 2810 DEFPROCpaldisp
 2820 PROCclearwind:VDU30:VDU14
 2830 PRINT"Filename: "oldpfile$
 2840 FOR col=0 TO 15
 2850 dummy=FNreadcolour(col)
 2860 PRINT"Col ";col TAB(8)":"TAB(15);r
ed TAB(20);green TAB(25);blue
 2870 NEXT
 2880 VDU15:PROCspaceget
 2890 PROCcewindow
 2900 ENDPROC
 2910 :
 2920 DEFFNselect(X1,X2,Y1,Y2)
 2930 =X>X1 AND X<X2 AND Y>Y1 AND Y<Y2
 2940 :
 2950 :--------------------------------
```

Listing 2

```
   10 REM          >ScrManDem4
   20 REM Program   Screen Manager Demo
   30 REM Version   A 0.3
   40 REM Author    Lee Calcraft
   50 REM RISC User April 1988
   60 REM Program   Subject to copyright
   70 :
   80 ONERROR ON:COLOUR7:REPORT:PRINT" a
t line ";ERL:END
   90 MODE12:OFF
  100 PROCcolours
  110 PROCtitle
  120 PROCstartbox
  130 PROCsmartbox
  140 PROCscreenman
  150 END
  160 :
  170 DEFPROCcolours
  180 COLOUR 8,0,0,64
  190 COLOUR 9,192,192,192
  200 COLOUR 10,144,144,144
  210 COLOUR 11,80,80,80
  220 COLOUR 12,32,176,240
  230 COLOUR 13,32,112,192
  240 COLOUR 14,240,144,128
  250 COLOUR 15,240,240,112
  260 VDU19,0,24,64,80,192
  270 ENDPROC
  280 :
  290 DEFPROCtitle
  300 GCOL13:RECTANGLE FILL 792,912,400,
80
  310 GCOL12:RECTANGLE FILL 780,920,400,
80
  320 VDU5:GCOL0
  330 MOVE 830,994:PRINT"RISC USER"
  340 MOVE 930,952:PRINT"SCREEN MANAGER"
  350 VDU4:OFF
  360 ENDPROC
  370 :
  380 DEFPROCstartbox
  390 Z=0
  400 FOR X=160 TO 480 STEP 320
  410 H=1
  420 FOR Y=912 TO 0 STEP -128
  430 PRINTTAB(5-20*(Z>7),H);Z
  440 H+=4:GCOL Z
  450 RECTANGLE FILL X,Y,160,96
  460 Z+=1
  470 NEXT:NEXT:ENDPROC
  480 :
  490 DEFPROCsmartbox
  500 PROClosenge(850,612,280,280,40,9,1
0,11)
  510 VDU5
  520 GCOL8:MOVE 914,826:PRINT"3D TEXT"
  530 GCOL14:MOVE 910,830:PRINT"3D TEXT"
  540 GCOL8:MOVE 910,790:PRINT"ENGRAVED"
  550 GCOL10:MOVE 914,786:PRINT"ENGRAVED
"
  560 GCOL15:MOVE 910,740:PRINT"Colour 1
5"
  570 GCOL8:MOVE 910,700:PRINT"Colour  8
"
  580 GCOL7:COLOUR7
  590 VDU4:OFF:ENDPROC
  600 :
  610 DEFPROClosenge(X,Y,WX,WY,W,C1,C2,C
3)
  620 GCOL C1:RECTANGLE FILL X,Y,WX,WY
  630 GCOL C3:MOVE X,Y:MOVE X+WX,Y
  640 PLOT85,X+WX,Y+WY
  650 GCOL7:LINE X,Y+WY,X+W,Y+WY-W
  660 GCOL C2:RECTANGLE FILL X+W,Y+W,WX-
2*W,WY-2*W
  670 ENDPROC
```

RU

OFF TO A FLYING START

Flying Start is database package for the Archimedes transferred from the IBM PC. Mike Williams reports on the potential of this software.

If any micro is to succeed it needs certain categories of applications software. Word processors like ArcWriter and 1st Word Plus are one such; database systems, such as Flying Start II, are another. Like Logistix, which we reviewed in RISC User Issue 3, Flying Start II from Mitre Software is an import from the PC world. In contrast to Logistix, Flying Start II will at present only run on an Archimedes under the PC emulator, but the company claim that a native mode version (compiled to ARM assembler) will be available by June. In the course of preparing this review, I used Flying Start with version 1.09 of the emulator.

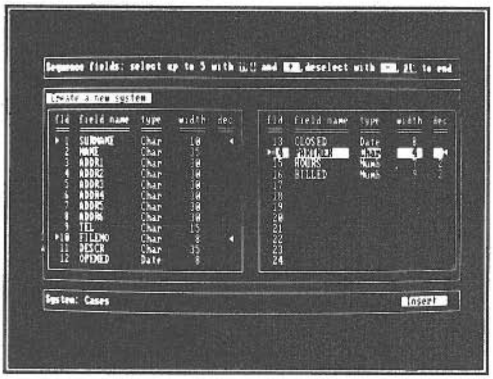
There are some further similarities with Logistix. Flying Start does not use a WIMP environment, and there are no pretty icons or menus. But Flying Start is marketed as a serious contender in the database stakes, and is thus none the worse for its more prosaic approach.

Let us examine what Flying Start has to offer. In essence, Flying Start provides a sophisticated form of 'card index' style database, with the extra facility to establish links between data files to form a simple relational database.

CREATING A DATABASE

The screen display has a fixed format, as in the illustrations, with a central working area, and single line displays at the head and foot of the screen to provide guidance, and reference information. Control of the system is effectively in the form of a hierarchical menu system, with a consistent use of a small set of keys for moving up or down the hierarchy and selecting options.

Flying Start enables you to create data files (called systems) with up to 24 fields, each with a maximum length of 256 characters. File size would appear to be limited only by disc capacity. Fields may be character, date, numeric or logical in type, and up to five fields may be selected as key (or index) fields for subsequent retrieval (and ordering) of records. As a consequence, no sort facility is provided or needed, and records are not maintained in any particular sequence. Whenever records are accessed, one of the key fields is first specified



```
Sequence fields: select up to 5 with [illegible] and [illegible], deselect with [illegible], [illegible] to end

Create a new system
 fld  field name  type   width  dec        fld  field name  type   width  dec
▶ 1   SURNAME     Char    10     ◀          13  CLOSED      Date     8
  2   NAME        Char    35              ▶[illegible] [illegible] [illegible] [illegible] [illegible]
  3   ADDR1       Char    30                15  HOURS       Numb     7     2
  4   ADDR2       Char    30                16  BILLED      Numb     9     2
  5   ADDR3       Char    30                17
  6   ADDR4       Char    30                18
  7   ADDR5       Char    30                19
  8   ADDR6       Char    30                20
  9   TEL         Char    15                21
▶10   FILENO      Char     8     ◀          22
 11   DESCR       Char    35                23
 12   OPENED      Date     8                24

System: Cases                                                      Insert
```

Creating a data file

and the records are retrieved in order according to that index. In practice records appear almost instantly using this concept, though I must add that time did not allow me to test response with very large data files. Furthermore, in reports (to screen or printer) records may be ordered by any field, part field or combination thereof by the automatic creation of a temporary index.

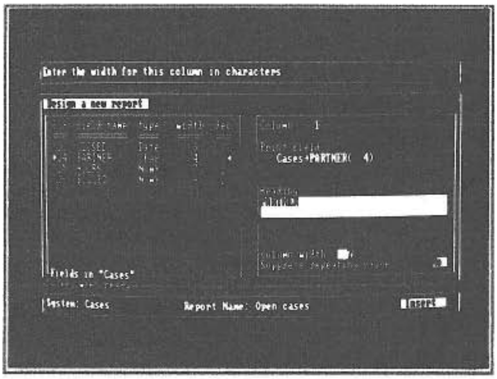
One feature I feel is significant is the facility to link data files together by means of a common (related) field. Thus a staff file containing names, addresses and other personal information may be linked to a file of hours worked by means of a common staff number. When a report is being prepared, fields may be included that come from the related file as well as from the primary file. Up to five secondary files may be related in this way to a single primary file.

REPORT GENERATION

It is no good storing masses of data in files if you cannot retrieve and present selected information in the form of reports. This activity forms the largest single section within the manual, and the options are quite comprehensive within certain constraints.

In common with the view of a Flying Start file as a card index (albeit one that may exist with up to five index orders), any report layout is based on a column format. A report may be given a two line report title, and each column may also be titled. You can also opt for the report to be dated and each page numbered, but you have no control over the position of either of these items.

In each column you select which field or fields are to be displayed, and whether whole or part fields. You can also specify a calculation as in a spreadsheet. The result of that calculation then appears in the specified column. As well as fields from the primary file, you can also include fields, as already mentioned, from any related file, and the left and right cursor keys allow you to switch between displays of the two sets of fields as the report layout is defined.



```
Enter the width for this column in characters

Design a new report
   Field name  Type   Width  Dec       Column    1
   [illegible] Date                    Print field
   PARTNER     Char     4                Cases+PARTNER(   4)
   [illegible] Num
   [illegible] Num                     Heading
                                       PARTNER

                                       Column width
                                       [illegible]
Fields in "Cases"

System: Cases            Report Name: Open cases          [illegible]
```

Generating a report

You can also exclude groups of records by selecting one of several operators (e.g. greater than, less than etc). This works well as far as it goes, but offers little flexibility. You can't match part strings for example, nor can you combine simple exclusion criteria with logical operators such as AND, OR or NOT. In fact, the use of exclusion, rather than a more positive inclusion, strikes me as curious. Once a report has been defined it may be saved for future use, or a report printed immediately. Reports can always be displayed on screen before printing.

A special form of report, labe printing, is catered for separately but similarly by Flying Start. Different label sizes and arrangements are provided for, and label formats may also be saved for use later. Both report and label designs may be recalled and modified at any time. Finally a number of options have been provided for the general maintenance of systems created. For example, files may be re-indexed, file structures modified and files linked or un-linked. Systems may be backed up, copied and transferred from disc to disc.

DOCUMENTATION

The documentation consists of two manuals, a tutorial manual and a reference manual. These are both clearly written and well produced. However, some illustrations are of rather poor quality and difficult to read. I also feel that not enough has been made of the sample files supplied on disc. These could have been used much more in the tutorial manual to explain the conçepts of Flying Start to the new user.

CONCLUSIONS

As a database system, Flying Start is well thought out and well presented. The use of indexes for file access and the facility to link files are real assets. The system is easy to use, with frequent use of default values (or responses) to speed things up. As a sophisticated card index system, Flying Start works well, but I am critical of the lack of greater flexibility with regard to record selection and report generation.

Compared with System Delta Plus (reviewed in RISC User Issue 3), I personally prefer Flying Start's simpler screen layout and menu system, and the manuals are infinitely superior. But System Delta Plus certainly has more features and offers greater flexibility in those areas where I have expressed some criticism of Flying Start.

Flying Start works well under the emulator, but I would suggest that it does not offer enough to justify the cost of the PC emulator for this purpose alone. If Flying Start satisfies your needs, it will perform admirably, and the advent of a native mode version should enhance its attractiveness considerably. It may not be as flashy as System Delta Plus, but it will do more than enough to meet the needs of many users just the same.

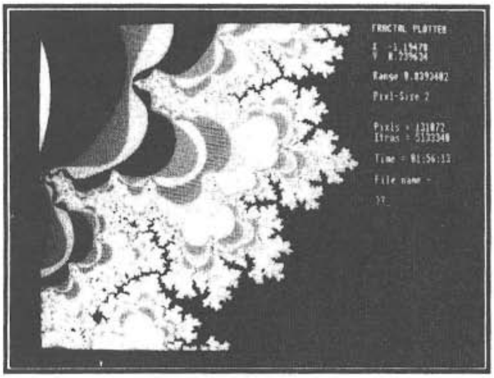
Product	Flying Start II
Supplier	Mitre Software Ltd, International House, 26 Creechurch Lane, London EC3A 5BA. Tel. 01-283 5614
Price	£69.95

RU

This month our 'Visuals' spot is provided by John Skingley with a program for displaying fractals (the Mandelbrot set), to produce a stunning and addictive display.

A FRACTAL PATTERN GENERATOR

The original motive for developing this program was to see just how fast my new computer would go. The answer was that it goes like the wind, and the graphics are superb. But in the course of discovering this, I've become hooked, and have definitely joined the 'Mandelbrot set'!



```
FRACTAL PLOTTER

X  -1.19478
Y   0.2396?4

Range 0.0392402

Pixl-Size 2

Pixls = 131072
Itras = 5133340

Time = 01:56:13

File name -

??_
```

Basically, the Mandelbrot set (named after Benoit B. Mandelbrot of IBM) is a set of solutions to a mathematical function, the one used here being the square function. To create the Mandelbrot set, take an initial value, apply the chosen formula to it, and then repeat using the result obtained as the new initial value. If we keep doing this, either the value will converge to some constant, or will increase, perhaps slowly at first then more rapidly, towards infinity. Whether it converges or not depends on the initial value. Starting values which converge belong to the Mandelbrot set. This set does not have a well defined boundary, but is 'fractal' in nature, providing us with the raw material for some fascinating patterns.

In order to produce two dimensional patterns, the screen area is used to map the complex plane, where each point represents an initial starting value for our function. Each point could be plotted in black if the function converged, and white if it sped off to infinity. However, far more intricate and beautiful patterns can be produced by choosing the colour of each point according to the speed with which the point speeds away. This is done by counting the number of iterations required for each starting point to increase past a critical value, beyond which it is destined for oblivion, and colouring the corresponding point according to the iteration count.

RUNNING THE PROGRAM

After typing in the program, save it to disc before testing. Assuming it is correct, the program will prompt for the initial starting values, in terms of the function used. Start by entering x=-2, y=-2, and range=4. This will give a screen representing the values -2 to +2 on both axes, and will display the complete Mandelbrot set for the function Zn = Z(n-1)^2 + u, where u is the starting value represented by the points on the screen.

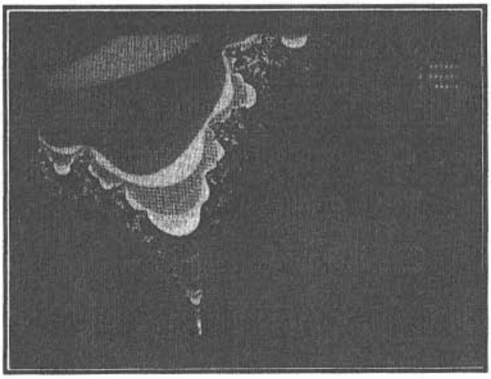
The program will also prompt for a pixel size. This facility is provided so that quick plots may be made using fewer screen points, while searching for the more interesting areas. The Archimedes is fast, but the highest resolution available in the mode used (15) requires over 130,000 points to be plotted, and each may take up to 64 iterations. One million iterations is not uncommon, and may take an hour or two! The pixel size is required in terms of screen units, so that a value of 2 or less will always produce the highest resolution (and take the greatest time). The displayed pixel is always square, and is best chosen from the binary sequence 2, 4, 8, 16 etc. I suggest starting with 16 or even 32.

Having entered this data, the fractal diagram will be plotted. When finished, the program will prompt for a filename for saving the screen to disc, which is performed using

*SCREENSAVE. This action is skipped if Return alone is pressed.

The mouse pointer is then displayed, and new starting values may be chosen by defining a rectangular screen area. Choose an interesting area at the edge of the black area, and click first one corner and then the diagonally opposite corner of this area. A moving rectangle is displayed during this operation.

At this point, the program will re-prompt for a pixel size for the new plot, which will then be drawn. In this way you may zoom in on interesting areas, and find that no matter how close you look, there is always more detail to be seen. The screen will always display the co-ordinates used, the iterations computed, and the time taken, for future reference. Escape will abandon any plot and return to the input stage; Escape at that point exits from the program.

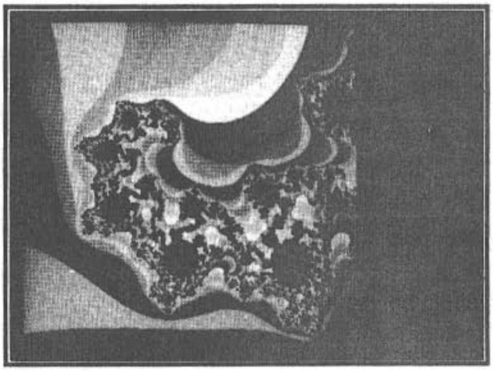


THE PROGRAM STRUCTURE

This is quite straightforward, the main work being done by PROCdraw. The actual function iteration is executed by the inner WHILE loop at lines 430-460. The procedure PROCmouse handles the input of the new co-ordinates from the screen, and translates these into new starting values for the function, while PROCinit computes the step sizes required by the FOR loops used to scan the screen, and initialises the points, iteration count, and the time. If you have not grasped the mathematics of all this, don't worry. Just run the program and enjoy exploring the Mandelbrot set faster than on any other micro.

SAMPLE SCREENS

The sample patterns included on this month's disc were made using a version of this program written in C. This resulted in a speed increase of over 2:1, so you will not match the times shown on these screens using the Basic version presented here.



```
 10 REM >Fractals
 20 REM Program   Fractal Generator
 30 REM Version   A 1.0
 40 REM Author    John Skingley
 50 REM RISC User April 1988
 60 REM Program  Subject to copyright
 70 :
 80 MODE 15:ON ERROR PROCerror:END
 90 sxmax%=1280:symax%=1024
100 xmax%=640:ymax%=256
110 xpix%=sxmax%/xmax%
120 ypix%=symax%/ymax%
130 @%=&00000700
140 :
150 REM Input initial data
160 PRINT"FRACTAL PLOTTER"
170 INPUT'"X orig ";xorig
180 INPUT"Y orig ";yorig
190 INPUT'"Range  ";range
200 INPUT'"Pixel-Size ";size%
210 :
220 REM Main program loop
230 REPEAT
```

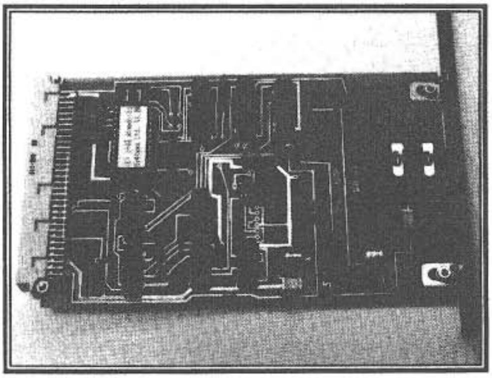
```
  240 CLS:PROCtext
  250 PROCinit:PROCdraw
  260 PRINTTAB(65,15);"File name -"
  270 ON:INPUT TAB(65,17);">";File$
  280 IF File$ <> "" THEN OSCLI "SCREENS
AVE "+File$
  290 PROCmouse
  300 UNTIL FALSE
  310 END
  320 :
  330 DEF PROCerror:MODE 12:REPORT
  340 PRINT" at line ";ERL:ENDPROC
  350 :
  360 DEF PROCdraw:LOCAL ERROR
  370 ON ERROR LOCAL ENDPROC
  380 OFF:a=xorig
  390 FOR j%=0 TO symax%-xstep% STEP xst
ep%
  400 b=yorig
  410 FOR k%=0 TO symax%-ystep% STEP yst
ep%
  420 x=0:y=0:n%=0:z=0
  430 WHILE z<4 AND n%<64
  440 xsqr=x*x:ysqr=y*y:z=xsqr+ysqr
  450 y=2*x*y+b:x=xsqr-ysqr+a:n%+=1
  460 ENDWHILE
  470 GCOL n%
  480 IF xstep%>xpix% THEN
  490 RECTANGLEFILL j%,k%,xstep%-xpix%,y
step%-ypix%
  500 ELSE PLOT 69,j%,k%
  510 ENDIF
  520 points%+=1:loops%+=n%
  530 time%=(TIME-start%)/100:b+=yinc
  540 NEXT
  550 PRINTTAB(65,10);"Pixels = ";points
%
  560 PRINTTAB(65,11);"Itrns = ";loops%
  570 PRINTTAB(65,13);"Time = ";FNtime(t
ime%)
  580 a+=xinc
  590 NEXT
  600 ENDPROC
  610 :
  620 DEF PROCinit
  630 xstep%=FNmax(xpix%,size%)
  640 ystep%=FNmax(ypix%,size%)
  650 xinc=xstep%*range/symax%
  660 yinc=ystep%*range/symax%
  670 points%=0:loops%=0
  680 start%=TIME
  690 ENDPROC
  700 :
  710 DEF PROCtext
  720 PRINTTAB(65,0);"FRACTAL PLOTTER"
  730 PRINTTAB(65,2);"X  ";xorig
  740 PRINTTAB(65,3);"Y  ";yorig
  750 PRINTTAB(65,5);"Range ";range
  760 PRINTTAB(65,7);"Pixel-Size ";size%
  770 ENDPROC
  780 :
  790 DEF PROCmouse
  800 MOUSE TO 512,512
  810 MOUSE RECTANGLE 0,0,1024,1024
  820 GCOL 3,7:*POINTER
  830 REPEAT:MOUSE x1%,y1%,B%:UNTIL B%
  840 REPEAT:MOUSE x2%,y2%,B%:UNTIL B%=0
  850 REPEAT
  860 MOUSE x2%,y2%,B%
  870 RECTANGLE x1%,y1%,x2%-x1%,y2%-y1%
  880 RECTANGLE x1%,y1%,x2%-x1%,y2%-y1%
  890 UNTIL B%
  900 MOUSE OFF
  910 xorig+=FNmin(x1%,x2%)*range/symax%
  920 yorig+=FNmin(y1%,y2%)*range/symax%
  930 range=(ABS(x1%-x2%)+ABS(y1%-y2%))*
range/(2*symax%)
  940 INPUTTAB(65,19);"Pixl-Size ";size%
  950 ENDPROC
  960 :
  970 DEF FNmax(a%,b%)
  980 IF a%>b% THEN =a% ELSE =b%
  990 :
 1000 DEF FNmin(a%,b%)
 1010 IF a%<b% THEN =a% ELSE =b%
 1020 :
 1030 DEF FNtime(t%)
 1040 LOCAL t$:t$="  :  :  "
 1050 MID$(t$,8,1)=STR$(t%MOD10):t%=t%DI
V10
 1060 MID$(t$,7,1)=STR$(t%MOD 6):t%=t%DI
V 6
 1070 MID$(t$,5,1)=STR$(t%MOD10):t%=t%DI
V10
 1080 MID$(t$,4,1)=STR$(t%MOD 6):t%=t%DI
V 6
 1090 MID$(t$,2,1)=STR$(t%MOD10):t%=t%DI
V10
 1100 MID$(t$,1,1)=STR$(t%MOD10)
 1110 =t$
```

armadillo audio sampler podule

Reviewed by Lee Calcraft

The Armadillo A448 sampler podule is one of the first of a range of devices to appear in the marketplace which exploit the Archimedes digital sound system. The unit consists of a half width podule, which plugs into the podule backplane, and a disc containing applications software.

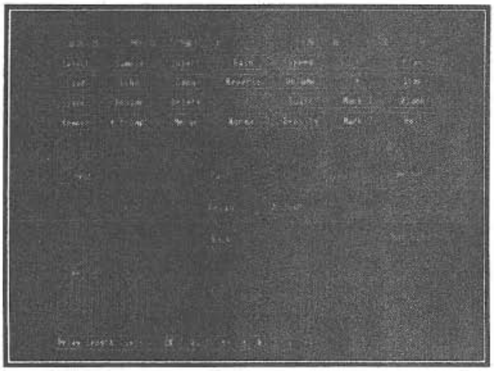
The podule is very easily fitted to the backplane, and once installed, it will automatically load its on-board "Sound Sampler" module into the computer's RMA. Once the podule is installed in the machine, the applications disc may then be booted. The package presents just a single four-colour screen (16 colours would have been better, and well worth sacrificing the extra 40K of screen RAM), which displays the rather bewildering mass of selection boxes shown in the accompanying picture. I found that this display, with its lack of hierarchy, was not easy to use, and that items very easily got lost in the sea of boxes.



The brief, though well-written, manual suggests that you start to experiment with the package by loading in the demonstration sample. This is accomplished by selecting ***LOAD,*** and supplying the filename. The waveform of the sound appears in the main window of the display, and you can look at various parts of this by selecting the left or right arrow boxes on the screen. Unfortunately, this causes the display to jump in screen-sized leaps in either direction rather than performing a sideways scroll; though Armadillo are considering improving this. Once a sound sample is in the machine, pressing the select button on the ***PLAY*** box causes the Archimedes to replay the sample - in this case, a few words from Armadillo.

MAKING YOUR OWN SAMPLES

This is very easily accomplished. First you must plug in a sound source to the podule's only socket (a standard mono jack socket). I used a simple dynamic microphone, but a wide range of sources is apparently acceptable. Then just select the ***SAMPLE*** box on the main screen. After you have supplied a sample name, you must acknowledge (and optionally adjust) the proposed parameters of sample length (2.5 secs) and sample rate (41 kHz). After you have made a few samples, this becomes tedious (the more so because the two **OK** boxes are not even at the same place).



The software should allow you to alter the time and rate if you wish, but should not insist that it is confirmed at the start of each recording. Moreover, the program should "remember" the user's previous choice of parameters rather than always insisting on its own. Next, a volume level meter is displayed. This responds to sounds input at the microphone, or wherever, and allows you to adjust the minute preset control at the back of the podule if necessary (no mean feat if you are trying to watch the screen and talk into the microphone at the same time!). Again I felt that the level indicator display could have been improved upon.

When you are happy with the level, press any key on the keyboard, and sampling begins. When the sample is complete, the graphical display still shows the previously selected sample (not very sensible), and you need to use the ***SELECT*** box to select the name of your latest sample before you can do anything with it. The pull-down menu option here also needs a bit more work doing on it. First of all, the pointer disappears when this box is used, which is a little disconcerting, and secondly, it always comes up with the first name in the list highlighted. It would be little trouble to highlight the *currently selected* sample from the outset.

Once you have made a few samples using the system, you will very rapidly run out of memory on a 300 series machine, because a 2.5 second sample made at the default rate of 41 kHz takes up around 100K of RAM. Unfortunately there is no display of memory used and memory remaining, so you have no real idea of what is going on until your request to load or create a further sample is turned down - again, the software could easily be improved here.

Apart from playing your samples, or saving them away to disc for later use, you can edit them in a number of interesting ways. Two vertical cursors are provided which allow you to perform a variety of cut and paste operations, making it very easy to produce effects such as the ubiquitous N-N-N-Nineteen. There is even a *REVERSE* button to reverse the order of the sampled data, and a special *ECHO* option which synthesises echo effects on any given sample. But once you have created some interesting sounds, what can you do with them? Armadillo are currently at work on a number of supplementary utilities. These include an option which will take a sample file, and parcel it up into a stand-alone relocatable module for use as a normal voice on the Archimedes sound system. You will then be able to play Grieg's Peer Gynt using parts of human voices, or any other sound that you can get your microphone to. When these utilities are completed, they will be offered as a free upgrade to existing users.

With their software smartened up a little (we understand that some of the criticisms made here will be taken into account in the release version of the software), the Armadillo package will offer an intriguing and useful tool to Archimedes owners interested in sound reproduction. However, intending purchasers should watch these pages for a review of a very similar product from Resource. Resource's new podule offers similar sampling facilities together with a 16 bit input and a 16 bit output port. It is due to be released in a few weeks from now.

The Armadillo A448 Sound Sampler costs £126.50 inc. VAT.
Armadillo are on (0572) 822 499.

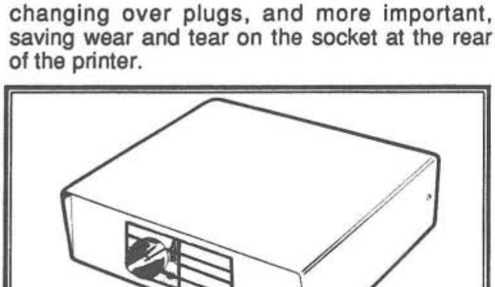
RU

THE TECHNOMATIC PRINTER SHARER BOX

Reviewed by Lee Calcraft

Technomatic's Printer Sharer Box allows you to connect two computers to a single parallel printer, or two parallel printers to one computer. This is ideal if you use a dot matrix printer for rough work, and a daisy wheel for high quality print. But even more relevant to Archimedes users is that this device allows you to switch your printer between Beeb and Arc without the necessity of changing plugs at the back of the printer.

The box is neatly finished in Acorn beige, with a single rotary switch on the front panel - and its two positions are conveniently labelled A and B (for Arc and Beeb obviously). At the rear are 3 Centronics parallel sockets. And it is this which gives the device its flexibility. All you need for printer sharing is a standard Centronics printer lead for each computer, and a Centronics to Centronics lead to go from the box to the printer. It all looks neat, and works very effectively, saving the hassle of repeatedly changing over plugs, and more important, saving wear and tear on the socket at the rear of the printer.



The Printer Sharer Box, which does not require batteries, is available direct from Technomatic, or from BEEBUG Retail. Technomatic's price for this unit is £28.75 inc. VAT. They are running a special offer to RISC User members of £24.15 providing that the order is received by 31 May.
Technomatic are on 01-208 1177.

RU

BBC to ARM Software Conversion

Lance Allison examines a suite of programs from Resource designed to make the process of upgrading from Beeb to Archimedes that much easier.

One of the claims made by Acorn for the Archimedes is its high level of compatibility with BBC machines. You may question the extent to which this is true when you actually sit down and try transporting software from one micro to the other. Acorn do provide a 6502 emulator on the Archimedes Welcome disc, and this includes a copy of BasicIV as used on the Master 128. However, there are a number of ways in which BBC programs may fail to work on an Archimedes, and it is this problem that the suite of programs reviewed here is designed to address.

The BBC-ARM Utilities from Resource comprises a single Archimedes formatted disc and accompanying manual. Reading the manual for the first time, I was struck by the technical manner in which it was written. The manual assumes that the reader is familiar with many technical terms, for which no explanation is offered.

The software on the disc falls into two categories. The first are 'straight' utilities, and include programs to convert BBC screen images, GXR sprites, and user defined graphics characters into a readable form on the Archimedes.

Secondly, there are two emulators included on the disc. The first is, surprisingly, the same 6502 emulator as supplied on the Archimedes Welcome disc. The second is an extended 6502 emulator which additionally provides DFS emulation, intercepts a number of Operating System calls, and deals with user-defined graphics characters.

TROUBLE-SHOOTING BBC PROGRAMS

The problems most likely to cause trouble when transferring BBC programs to an Archimedes, are any direct references to memory (as with indirection operators and the LOADing and *SAVEing of screens). A utility is provided which will check a Basic program for any such problems. All circumstances of indirection operators, and the use of various Basic keywords such as SOUND and ENVELOPE, will be flagged for attention. Unfortunately, much time is wasted, as many of the possible problems highlighted turn out to be perfectly acceptable on the Archimedes.

As mentioned previously, there are two programs for converting BBC screen files and GXR sprite files for use on the Archimedes. If you have written programs on a BBC, with sprites designed using the GXR ROM, you can convert the sprite data into the same format as the sprites used on the Archimedes. Another utility allows character definitions, stored at &C00 on the BBC to be salvaged (to quote the manual) and converted into VDU commands so that they can also be used on the Archimedes.

DFS EMULATION

The way that DFS emulation works is to treat a single ADFS directory as a single side of a DFS disc. This means that four ADFS directories can be defined as four separate DFS discs, and be addressed in the usual DFS manner. To do this, a command, *LINK, formats an ADFS directory so that it can be used for DFS emulation. This implementation of the DFS is very thorough and should allow any Basic programs written for the DFSto run.

CONCLUSION

This package provides a number of useful utilities for conversion of programs from the BBC Micro to the Archimedes, but do be aware of the highly technical nature of the manual. Remember, too, that you will still have to perform much of the conversion work yourself.

Product:	BBC-ARM Utilities
Supplier:	RESOURCE, Exeter Road, Doncaster DN2 4PY. Tel.(0302) 63800/63784
Price:	18.35 inc.VAT

REVIEWS

THE ACORN MIDI MODULE

Reviewed by Lee Calcraft

Last month we reviewed Acorn's I/O podule board. This has an optional add-on "MIDI module", which is the subject of the present review. It is important at the outset to distinguish this product from Acorn's dedicated "MIDI podule". The one letter difference in the name obscures an important performance and price difference, and we shall be reviewing the latter product in the near future.

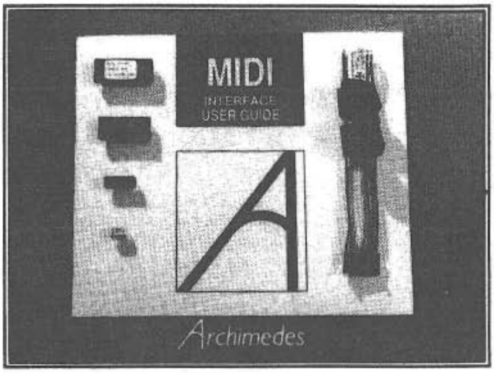
The MIDI module add-on consists of a serial interface chip, a couple of DIN sockets and an EPROM upgrade for the on-board I/O podule software. Once the unit is installed on the I/O podule board, it provides the user with an additional relocatable module called MIDI which provides around 30 dedicated SWI calls. At present the module comes with no user-friendly software, and Acorn hope that third party software houses will fill this gap. The manual which comes with the upgrade reflects this, and consists of little more than a programmer's guide to the SWI calls.

Having said this however, the calls are easy to use at an elementary level at least. To test out the module, I connected it up to a Yamaha RX21 drum machine. All this involves is connecting a MIDI cable between the MIDI in socket on the Yamaha and the MIDI out on the Arc. After putting the drum machine into "MIDI receive" mode, all you need to do is to execute the appropriate SWI to send to the instrument. To show how easy this is, the following will activate the Side Drum:

```
    SYS "MIDI-TxNoteOn",52,64
```

The first of the two parameters here is the note to be played, and the second is the so-called "key on velocity", often simply interpreted as the volume. It ranges from 0 to 127.

On the RX21, each of its nine instruments can only play at a fixed pitch, and each has therefore been assigned a unique "note number". Note 60, which would normally play a middle C, plays the cymbals, note 59 the open high hat, and so on. To show how easy it is to control the drum machine from the Arc, I have included a short program which generates a 12 beat repeated rhythm. It would be a relatively simple matter to write a user-friendly sequencer for the RX21.



```
   10 REM >Midi2
   20 DIM inst(100):PROCinit
   30 REPEAT: FOR A=1 TO N-1
   40 IF inst(A)>0 SYS "MIDI_TxNoteOn",i
nst(A),127
   50 Z=INKEY(20):NEXT:UNTIL FALSE
   60 :
   70 DEFPROCinit
   80 N=0:REPEAT:N+=1:READ inst(N)
   90 UNTIL inst(N)=1000:ENDPROC
  100 DATA 52,0,54,54,0,54,57,57,59,0,57
,0,1000
```

EXTENDING THE SYSTEM

The MIDI module is quite capable of simultaneously controlling more than one MIDI instrument. This is achieved by altering the transmit channel (the default is channel 1), using SYS"MIDI_SetTxChannel". You should also be able to use the Archimedes as an "instrument", and control its 8 voices from a remote MIDI keyboard, but this is unfortunately not possible on the MIDI module, because the serial interface is not fast enough. The MIDI podule will however be capable of this, and users who need this dual capability would be advised to wait for its forthcoming release. Otherwise, notwithstanding the lack of application software and technical nature of its manual, this is a most attractive product.

Acorn's MIDI add-on module costs £33.35 Inc. VAT.
Acorn are on (0223) 214411.

Thanks to Rob Barnes for the loan of the RX21.

RU

INTRODUCING ARM ASSEMBLER

by Lee Calcraft

ARM machine code is the fastest micro-computer language in the world. It is also an interesting and very rewarding language to get to grips with, as we hope you will discover from this short series.

Although I shall begin this series with some brief notes about ARM assembler, these articles will be practically orientated, and we shall quickly move on to specific examples which will impart the flavour of the beast much more effectively than a blow by blow account of each instruction. If you wish to read about ARM assembler in parallel to these articles, I would recommend the only book on the subject: *ARM Assembly Language Programming* by Peter Cockerell. It is both thorough and well written, though with insufficient illustrative examples for my liking.

ARM ASSEMBLY LANGUAGE

ARM machine code is the language of the ARM (Acorn RISC Machine) central processor, and ARM Assembly language is a mnemonic-based language in which each instruction corresponds to a single ARM machine code instruction. As with earlier BBC Micros, Acorn have thoughtfully implemented a full assembler within Archimedes Basic, though this is only briefly documented in the *Programmer's Reference Manual*, and not at all in the *User Guide.*

ARM FUNDAMENTALS

If you are graduating to the ARM from the 6502 or other 8-bit processors, you will be delighted at what it has to offer. Instead of the three 8-bit user registers of the 6502, the ARM provides fifteen 32-bit registers all at the user's full disposal. It also has a number of other registers used at so-called supervisor level. Each instruction is 32 bits in length, and instructions must be located at 32-bit word boundaries, though the language can easily handle byte-wide data as well.

One way in which the ARM achieves its speed is to keep its instruction set down to a minimum. Another is its extensive use of the technique called "pipelining". In the case of the ARM this means that while it is performing any given instruction, it is simultaneously decoding the next, and fetching the one after that. This is very clever stuff, which has an implication for the way in which programs are written because every time that a program performs a branch instruction the next two instructions in the pipeline must be scrapped (because the program has branched elsewhere), with an obvious time overhead.

Before delving into the language itself, there is one impression which should be dispelled. The acronym RISC stands for *Reduced Instruction Set Computer*, and implies that the ARM processor has a limited number of base instructions. Although this is true, all instructions can have one or more additional parts to them, giving rise to a vast number of different permutations. For example, all instructions can be made conditional. Thus while

```
ADD R0,R1,R2
```

means "add the contents of R2 to R1 and place the result in R0", the ARM chip is just as happy with:

```
ADDNE R0,R1,R2
```

which means that the addition will take place only if the zero flag is not set (i.e. ADD IF Not Equal). There are 16 such conditions, as we shall see later. To perform a similar operation in 6502 assembler would require something like:

```
BNE skip
LDA R1
CLC
ADC R2
.skip
```

The single ARM instruction which performs the same task thus avoids the need for a branch instruction, with its inherent time overhead due to the loss of pipelining.

But the power of the ARM does not end here. The operands of certain of its instructions can involve a further stage of complexity. For example, the ADD instruction can take the form:

```
ADD R0,R1,R2,LSR #16
```

This means that the contents of register R2 will be logically shifted right by 16 bits before being added to R1. And there is much more besides, as we shall see in the course of this series.

A SIMPLE INSTRUCTION TESTBED

The program in listing 1 performs a 32-bit addition. If you type it in and run it, you will be asked for two numbers, and their sum will be displayed as a result. The numbers supplied, and indeed the result, may be negative, but should not exceed 31 bits in length (a value of around 130 million). The top bit of the 32-bit word is used as a sign bit following the standard signed integer convention (try PRINT &FFFFFFFF).

Let us now take a look at what is going on. At the heart of the program is the sequence from line 40 to 210. The Archimedes uses the same assembler format as on earlier BBC micros, in which the assembly program is enclosed in square brackets (lines 70 and 200). We have used a so-called two pass assembly set up with the FOR-NEXT loop between lines 50 and 210. Lines 40 and 60 allocate an area of RAM into which the code will be assembled. Again this is the same as on earlier BBC micros, except that it is not advisable to specify memory addresses directly.

On the BBC micro, machine code is often assembled at directly specified locations - &A00 is often used for example. But this is not a reliable technique on the Arc, since the allocation of logical RAM is completely under operating system control, and can vary depending on configuration settings. RAM allocation for the assembler should therefore be reserved using a DIM statement similar to that in line 40. Here we have reserved a 200 byte block of RAM, and Basic places the actual start address of this block into the variable *space*. In assigning RAM in this way, you should remember that each ARM instruction takes *four* bytes, so a 200-byte block will provide sufficient space for 50 instructions. One further point: I mentioned earlier that each ARM instruction must be aligned to a 32-bit word boundary. The variable *space* returned from the DIM statement in line 40 is automatically aligned in this way.

Let us now take a look at the program proper. The input and output of data is handled in Basic between lines 230 and 300. Essentially the two numbers supplied by the user are placed into memory as 32-bit words (lines 260 and 270), and after the machine code is called (with the CALL statement at line 280), the result is taken from memory using the indirection operator at line 290.

The machine code consists of just five instructions. The first two, at lines 100 and 110, use the **LDR** (LoaD Register) instruction to load registers R1 and R2 with the two numbers to be added. Line 120:

```
    ADD R0,R1,R2
```

adds R2 to R1 and places the result into R0 (to help remember the order of parameters, think of the sum R0=R1+R2). Then at line 130 the **STR** instruction (STore Register) stores the result at location *table*. This is all easy stuff.

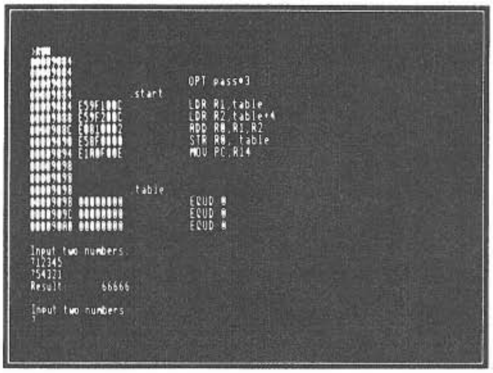
The final instruction is a little more complex:

```
    MOV PC,R14
```

This instruction MOVes the contents of R14 into the register known as **PC.** This is the program counter which holds the address of the next instruction, plus the ARM's flags (i.e. for those familiar with the 6502, it is a combination of the program counter and status register). It is in fact register R15, though it is a good idea to use the mnemonic **PC** because it will remind you that it is a very special register, and should not be utilised without due care and attention.

So the final instruction of our sequence moves the contents of R14 into the program counter. The effect of this is similar to the old **RTS** (or ReTurn from Subroutine) of the 6502. This works because when Basic uses the CALL statement to run a piece of machine code, it executes a so-called *Branch with link*. This branches to the appropriate address, and places the return address (in this case the address which will get it back into the Basic program) into register R14 (sometimes called the *link* register). Thus to return to Basic, we simply need to reinstate the PC with MOV

PC,R14. Moreover, because the PC also contains the status register, all of the processor's flags are automatically reinstated.



```
>RUN
00009084
00009084
00009084                  OPT pass*3
00009084        .start
00009084 E59F100C         LDR R1,table
00009088 E59F200C         LDR R2,table+4
0000908C E0810002         ADD R0,R1,R2
00009090 E58F0000         STR R0, table
00009094 E1A0F00E         MOV PC,R14
00009098
00009098
00009098        .table
00009098 00000000         EQUD 0
0000909C 00000000         EQUD 0
000090A0 00000000         EQUD 0

Input two numbers:
?12345
?54321
Result:      66666

Input two numbers:
?
```

SUBTRACT AND MULTIPLY

The testbed program in listing 1 will just as easily test out the ARM's subtract and multiply instructions. To achieve this, replace line 120 as follows:

```
      120 SUB R0,R1,R2   to subtract
or    120 MUL R0,R1,R2   to multiply
```

and enter two numbers as before in response to the prompts. Again the only proviso is that the 31-bit limit is not exceeded. In a similar way you can also test the three logical instructions EOR, ORR and AND. These perform exclusive OR, OR and AND operations, and use a similar syntax. Thus:

```
120 EOR R0,R1,R2
120 ORR R0,R1,R2
120 AND R0,R1,R2
```

THE CMP INSTRUCTION

Finally we can use listing 1 to test out the CMP instruction. This behaves very similarly to its 6502 counterpart. To try it out, alter listing 1 as follows:

```
120 CMP R1,R2
130 STRLT R2,table
135 STRLT R1,table+4
290 PRINT"Result: "!table,!(table+4)
```

If you now run the program, you will see that the result echoes the two numbers input at the prompt, and that they are now always displayed in order of magnitude.

This ordering of the data is achieved using the CMP instruction in line 120. It compares the contents of registers R1 and R2, and sets the processor's flags accordingly. The two instructions which follow at lines 130 and 135 are made conditional on the results of this comparison. The **LT** at the end of the two mnemonics means that these two instructions will only take place if R1 was *Less Than* R2 in the CMP instruction above. Thus the two numbers stored at *table* and *table + 4* are only swapped over if the second is larger than the first. This has all been achieved without recourse to branching, thus keeping the code fast and simple. I hope that it gives a hint at least of the power of the ARM instruction set.

Next month we will take a further look at the condition mnemonics, and introduce some more new instructions.

Listing 1

```
 10 REM >TestAbed
 20 REM Test 32 bit addition
 30 :
 40 DIM space 200
 50 FOR pass=0 TO 1
 60 P%=space
 70 [
 80 OPT pass*3
 90 .start
100 LDR R1,table
110 LDR R2,table+4
120 ADD R0,R1,R2
130 STR R0, table
140 MOV PC,R14
150 :
160 .table
170 EQUD 0
180 EQUD 0
190 EQUD 0
200 ]
210 NEXT
220 :
230 REPEAT
240 PRINT'"Input two numbers:"
250 INPUT A,B
260 !table=A
270 !(table+4)=B
280 CALL start
290 PRINT"Result: "!table
300 UNTIL FALSE
```

RU

ALL-MODE SCREEN COMPRESSOR

by Mark Davis

Use this invaluable screen compressor to massively decrease screen saving and loading times, and to save on valuable disc space.

As you will probably be aware, screen saving and loading using the Arc's built-in commands is a slow process. And although the FastSave and FastLoad routines published in RISC User Issue 3 solve this problem, they do nothing to economise on disc space. Archimedes screens are very hungry on disc space - so much so that you can only squeeze nine mode 12 screens, or four mode 15 screens on an empty 800K disc.

The listing presented here is an attempt to remedy the problem. It provides the user with two procedures, one for compressing and saving a screen in any mode, and another for loading it back in and expanding it. As with all screen compression routines, the degree to which any given screen can be compressed depends on the degree of redundancy which it contains. The algorithm used here is particularly effective in the 256 colour modes, and to give an example, it saved a complex mode 15 Mandelbrot screen in just 7.8 seconds, reloading it in a mere 3.4 seconds. This compares very favourably with the 37 seconds taken by *ScreenLoad. Moreover, the compacted screen took up just 48K of disc space. This is less than one third of the space used by the original screen.

USING THE PROGRAM

First of all, type in the listing, and save it away to disc. To put it to use you will need to append it to any program which creates or makes use of screen displays. You can do this using the APPEND command or LIBRARY or INSTALL. As an example, take a look at the following:

```
10 PROCinit:*ScreenLoad Screen
20 PROCsave("CScreen"):IF GET
30 PROCload("CScreen")
40 END
```

The first line calls PROCinit to set up the save/load facility. It then uses *ScreenLoad to load in a normal screen named "Screen". Next, PROCsave is called to save it in compressed form under the name "CScreen". Finally, if any key is pressed, the compressed screen will be loaded in again and expanded.

To try this out, just type in the above then:

`APPEND "CompressF"` (or whatever)

Now RUN the combined program. Alternatively, to display the screen in another program, call PROCinit to set up the code, then PROCload to load in and expand the screen.

PROGRAM NOTES

1. PROCinit should only be called once in any program.
2. PROCsave must *always* be called from a non-scrolled or freshly cleared screen. Note that *ScreenLoad automatically performs a screen clear.
3. Be careful to avoid conflicts of variable names with your main program.
4. Remember that the compressor claims 160K of user RAM as workspace.
5. If a particular screen cannot be compressed, it will still be saved at high speed, but the length of file created will be the same as for *ScreenSave. This is most likely to occur in mode 12.

```
   10 REM            >CompressF
   20 REM Program    Screen Compression
   30 REM            Procedures
   40 REM Version    A 1.0F
   50 REM Author     Mark Davis
   60 REM RISC User  April 1988
   70 REM Program    Subject to Copyright
   80 :
   90 DEFPROCinit
  100 DIM block 2000,final &28000
  110 block2=block+256:code=block+512
  120 DIM sbl 100:CLOSE #0
  130 FOR PASS=0 TO 2 STEP 2
  140 P%=code
  150 [ OPT PASS
  160 .no EQUD0
  170 .scaddr EQUD0
  180 .sclen EQUD &27FFF
  190 .fin EQUD final
  200 .shift EQUD0
  210 .max EQUD0
  220 .total EQUD0
  230 .b EQUD 148:EQUD7:EQUD-1
  240 .b2 EQUD0:EQUD0:EQUD0
  250 .init
  260 STMFD R13!,{R14}
  270 ADR R0,b:ADR R1,b2:SWI &31
  280 LDR R0,b2:STR R0,scaddr
  290 LDR R0,b2+4:SUB R0,R0,#1
  300 STR R0,sclen:ADR R0,block2
```

```
310 MOV R1,#64:MOV R2,#0:.inlp
320 STR R2,[R0],#4:SUBS R1,R1,#1
330 BNE inlp:LDMFD R13!,{R15}
340 .compact
350 STMFD R13!,{R14}
360 BL init:BL search:CMP R11,#8
370 MOVCS R0,#1:LDMCSFD R13!,{R15}
380 LDR R8,scaddr:LDR R9,sclen
390 LDR R7,fin
400 .comlp
410 LDRB R0,[R8],#1:BL check
420 MOV R6,R11
430 MOV R2,#0:LDR R12,max
440 .nlp LDRB R1,[R8],#1:CMP R1,R0
450 ADDEQ R2,R2,#1:SUBNE R8,R8,#1
460 BNE enext:SUBS R9,R9,#1
470 BEQ enext:CMP R2,R12:BCC nlp
480 .enext
490 LDR R11,shift
500 ADD R11,R6,R2,ASL R11
510 STRB R11,[R7],#1
520 SUBS R9,R9,#1:BPL comlp
530 STR R10,no:STR R7,total
540 MOV R0,#0:LDMFD R13!,{R15}
550 .search
560 STMFD R13!,{R14}
570 MOV R10,#0:STR R10,no
580 LDR R8,scaddr:LDR R9,sclen
590 LDR R7,fin
600 .sealp
610 LDRB R0,[R8],#1:BL check
620 CMP R12,#0:ADREQ R14,block2
630 STREQB R10,[R14,R0]:ADD R2,R2,R10
640 STREQB R0,[R2],#1:ADDEQ R10,R10,#1
650 SUBS R9,R9,#1:BNE sealp
660 STR R10,no:MOV R11,#0
670 MOV R12,#256:MOV R0,#1
680 .schshflp
690 ADD R11,R11,#1:MOV R12,R12,LSR#1
700 CMP R10,R0,ASL R11:BCS schshflp
710 STR R11,shift:SUB R12,R12,#1
720 STR R12,max:LDMFD R13!,{R15}
730 .decompact
740 STMFD R13!,{R14}
750 BL init
760 LDR R8,scaddr:LDR R9,sclen
770 LDR R7,fin:ADR R2,block
780 LDR R11,shift:MOV R12,#&FF
790 .dcomlp
800 LDRB R3,[R7],#1
810 BIC R1,R3,R12,ASL R11
820 ADD R4,R2,R1:LDRB R0,[R4]
830 MOV R5,R3,LSR R11
840 .dclp2 STRB R0,[R8],#1
850 SUB R9,R9,#1:SUBS R5,R5,#1
860 BPL dclp2:CMP R9,#0:BGT dcomlp
870 LDMFD R13!,{R15}
880 .check
890 STMFD R13!,{R14}
900 ADR R2,block:CMP R10,#0
910 MOVEQ R12,#0:MOVEQ R11,#0
920 LDMEQFD R13!,{R15}
930 ADR R14,block2:LDRB R11,[R14,R0]
940 LDRB R3,[R2,R11]:CMP R3,R0
950 MOVEQ R12,#1:MOVNE R12,#0
960 LDMFD R13!,{R15}
970 ]:NEXT:ENDPROC
980 :
990 DEFPROCsave(F$)
1000 C%=USR(compact)
1010 Q%=OPENOUT(F$):BPUT#Q%,MODE
1020 BPUT#Q%,C%:BPUT#Q%,!no
1030 FOR X=0 TO 15:?sbl=X:SYS 7,&B,sbl
1040 BPUT#Q%,sbl?1:BPUT#Q%,sbl?2
1050 BPUT#Q%,sbl?3:BPUT#Q%,sbl?4
1060 NEXT
1070 SYS &C,2,Q%,block,256
1080 BPUT#Q%,!shift:BPUT#Q%,!max
1090 A%=!total-!fin
1100 BPUT#Q%,A%:BPUT#Q%,A% DIV&100
1110 BPUT#Q%,A% DIV&10000
1120 BPUT#Q%,A% DIV&1000000
1130 IF C%=0 THEN
1140 SYS &C,2,Q%,!fin,!total-!fin
1150 ELSE
1160 SYS &C,2,Q%,!scaddr,!sclen+1
1170 ENDIF:CLOSE#Q%
1180 ENDPROC
1190 :
1200 DEFPROCload(F$)
1210 Q%=OPENIN(F$):VDU22,BGET#Q%
1220 C%=BGET#Q%:!no=BGET#Q%
1230 FOR X=0 TO 15:?sbl=X
1240 sbl?1=BGET#Q%:sbl?2=BGET#Q%
1250 sbl?3=BGET#Q%:sbl?4=BGET#Q%
1260 SYS 7,&C,sbl:NEXT
1270 SYS &C,4,Q%,block,256
1280 !shift=BGET#Q%:!max=BGET#Q%
1290 A%=BGET#Q%+(&100*BGET#Q%)+(&10000*
BGET#Q%)+(&1000000*BGET#Q%)
1300 !total=A%+!fin
1310 IF C%=0 THEN
1320 SYS &C,4,Q%,!fin,!total-!fin
1330 ELSE
1340 SYS &C,4,Q%,!scaddr,!sclen+1
1350 ENDIF:CLOSE#Q%
1360 IF MODE=2 OR MODE=9 OR MODE=12 THE
N *FX9
1370 IFC%=0 THEN CALL decompact
1380 ENDPROC
```

REVIEWS

THE BEEBUG 5.25 INCH DRIVE INTERFACE

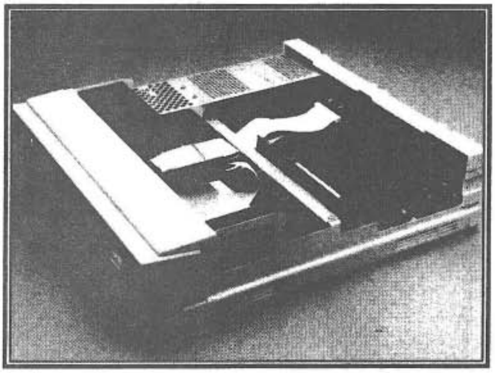
Reviewed by Dennis Weaver

It is not possible to connect an external 5.25 inch disc drive directly to the Archimedes, because the Arc's disc interface is insufficiently buffered. The new drive interface from BEEBUG makes this possible. It is supplied as a small board which is mounted inside the Arc, taking the position of half of a single back panel (though it does not require a backplane socket, and leaves room for a backplane populated with one full-width and one half-width podule).

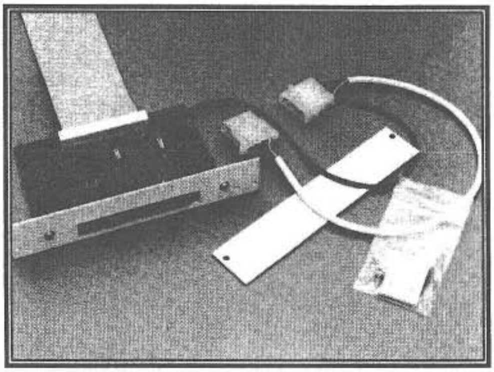
Installing the unit is relatively easy, and although it involves no soldering, you do need to replace the Arc's internal drive lead with the one supplied (which is suitable for systems with either one or two internal floppies). When the unit has been installed, you will need to use:

```
    *CONFIGURE FLOPPIES 2    (or 3)
```

to tell the ADFS how many floppy drives are attached. You then simply plug your external disc drive into the new socket on the back panel of the Arc.



Well almost. You must first make sure that the external drive is allocated to an unused drive number, otherwise the Arc will attempt to access two drives at once. Assuming that you have a dual external drive, and only a single internal drive, you can ignore the problem. By default, the second of your twin drives will appear as Arc drive 1. If you have either two internal drives, and/or only one external drive, you will need to open your disc drive case and set it to an unused drive number (e.g. 2). This makes for problems if you want to repeatedly swap your drive between the Arc and the Beeb, since the Beeb cannot cope with drives numbered higher than 1.



For this review, I used a single Tandon drive from Viglen. Inside I altered the plug link from "DS0" to "DS2", and on plugging it into the Arc it worked first time. Using the RISC User ADFS menu worked like magic, and I was able to load, save and copy to and from the new drive, just as if it were an internal drive. Of course in many ways this obviates the need for Beeb to Arc transfer software, because you can operate directly from Beeb discs. You can also format them to 800K density (if they are high density discs), and it provides a way of using all those old floppies that litter the house. You should note however that even with this unit installed, the Archimedes cannot read DFS format discs. You will need to convert them to ADFS format first.

One further plus of this interface is that if you use the Archimedes PC Emulator, you will actually be able to obtain software for it. The majority of PC software is on 5.25 inch disc, and the Emulator works perfectly with the Arc external disc interface. If you need to run PC protected software which will only run from drive "A", the BEEBUG board has a special link which may be set to make the external floppy appear as drive 0.

This is a most useful piece of kit.

The 5.25 inch Drive Interface costs £29.05 inc. VAT (£27.60 to members).
BEEBUG are on (0727) 40303.

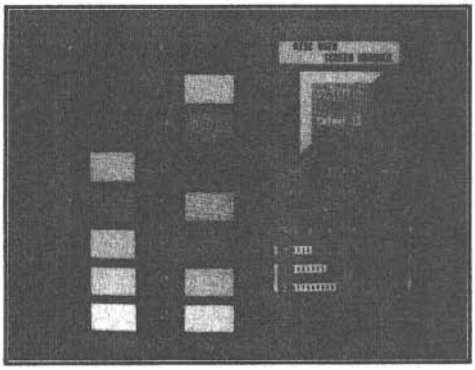
RU

RISC User Magazine Disc

April 1988

CONTENTS

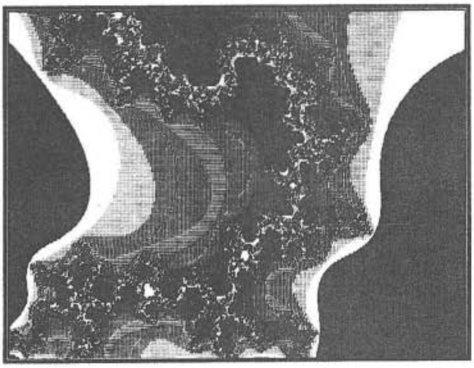
SCREEN COMPRESSOR - a highly useful utility which allows screen dumps to be compressed before saving to disc, and expanded when reloaded. This saves on disc storage and is faster than the standard SCREENSAVE and SCREENLOAD commands.



SCREEN MANAGER - is another versatile utility, this time for screen design and layout. It allows all the colours on the screen to be edited in context to achieve the best possible effects. Other screen information is also provided.

A WINDOW ON THE ARCHIMEDES - is a demonstration of the use of the Archimedes WIMP manager in action, and the start of a series of articles on this subject which will teach you all you need to know for effective window displays.

ARCHIMEDES VISUALS - in this issue is devoted to a single program which generates Mandelbrot graphics, a fascinating microcosm in all the Archimedes range of colours. We have also included a rolling display of some of the fascinating Mandelbrot displays we have generated.



A COCKTAIL OF 3D PROCEDURES - provides an example of the very smooth animation of wireframe objects achievable on the Archimedes. Again a program to use and learn by.

ARM ASSEMBLER - everyone will want to know about this subject and our new series on this subject starts off with two complete demonstration programs.

ADDITIONAL ITEMS

PERCUSSION MODULE - a relocatable module containing a set of no less than 13 different percussion instruments. These include a variety of different drums, cymbals, woodblocks etc. The normal SOUND command gives full control over volume, stereo position and channel. A short demonstration program is also provided.

3D ANIMATION - Acorn has given us permission to include on the RISC User magazine disc an amazing 3D animation. This consists of 50 separate images (files) so we have had to divide these into two, part this month and part next. We can assure you that the overall result will be a stunning demonstration of the Archimedes' speed and power.

All RISC User magazine discs are available to order, or by subscription. Full details of prices etc are given on the back cover of each issue of RISC User.

Basic V

A CASE OF BASIC V

In an occasional series on the new features of Basic V, Mike Williams explores the use of the CASE statement.

Although BBC Basic has always been one of the more structured Basics around, until the advent of Basic V on the Archimedes there were still many omissions. Prominent among these was a CASE statement, and in this short article I want to discuss the forms and uses of this statement as found on the Archimedes.

First of all, just what is a CASE statement? One way of understanding it is to think of it as a multiple IF-THEN-ELSE construction. This latter tests a condition, and depending upon whether the result is true or false allows the choice of two alternative courses of action. A CASE statement deals with the situation where a condition may have more than two states (or values), with a different course of action resulting from each.

To understand the construction used in Basic V let's consider a simple example. Suppose we have a short visual program like last month's Circles in which the mouse *select* button displays a coloured disc on the screen, the *menu* button increases its size, and the *adjust* button decreases its size. One way of coding this would be to write:

```
MOUSE x,y,z
CASE z OF
   WHEN 4: PROCdisplay
   WHEN 2: PROCincrease
   WHEN 1: PROCdecrease
ENDCASE
```

In the above, the value of z determines which of the mouse buttons has been pressed. Cgg the keyword CASE you specify the variable or expression whose value determines which course of action to follow, ending the line with the keyword OF. Following the CASE statement itself, there should be one or more WHEN statements. Each WHEN is followed by a particular value terminated by a colon, and then one or more statements which result in that case. In the example, a value of 4 indicates the left-hand button only, a value of 2 the middle button only, while 1 indicates the pressing of the right-hand button only. The list of WHEN statements is terminated by ENDCASE.

There is one important point to note in the interpretation of this structure. Basic searches through each WHEN until it finds a match for the current value of z (in the example). It then executes the statement(s) following that WHEN, jumping directly after to the first statement following ENDCASE. Thus only ONE of the WHEN statements will be matched on any one occasion, even if further matches do exist in later WHEN statements. This can mean that the order of the WHEN statements will be significant in some cases.

In the example, the coding picks out those cases where *only* the specified button has been pressed. Suppose we wish to modify the routine to act in cases where other buttons have been pressed as well:

```
MOUSE x,y,z
CASE z OF
   WHEN 2,6: PROCincrease
   WHEN 1,5: PROCdecrease
   WHEN 4  : PROCdisplay
   WHEN 3  : PROCclear
   WHEN 7  : PROCexit
ENDCASE
```

When z equals either 2 or 6 (equivalent to z=2 OR z=6) then PROCincrease is called, and similarly in the second case. Alternative values following WHEN are separated by commas. A match with any one will be accepted. Remember that these are simple examples, and the CASE variable, and the WHEN values may be complex expressions if necessary.

In addition to WHEN statements you can also include an OTHERWISE to cater for all other (unspecified) cases. There is no colon this time (no values to terminate) and it is simply followed by the relevant statements. Our original mouse example could be written as follows:

```
MOUSE x,y,z
CASE z OF
   WHEN 4: PROCdisplay
   WHEN 2: PROCincrease
   WHEN 1: PROCdecrease
   OTHERWISE VDU7
ENDCASE
```

Now, pressing any button combination other than one of the three buttons on its own will produce a (warning) bleep. The OTHERWISE, if used, should be the last statement before the ENDCASE.

CASE statements can also be nested - i.e. any statement following a WHEN may itself be a further CASE statement with its subsequent WHEN, OTHERWISE and ENDCASE. Such constructions can easily get out of hand so I suggest you keep things reasonably simple, and also lay out the code to help make the meaning clear. Here is an indication of what might be coded:

```
MOUSE x,y,z
CASE z OF
   WHEN 4: CASE POINT(x,y) OF
         WHEN red%:   PROCincrease(1)
         WHEN blue%:  PROCincrease(2)
         WHEN green%: PROCincrease(4)
         ENDCASE
   WHEN 2:                 (etc)
```

There is one special form of the CASE statement which appears not to be documented by Acorn. This is CASE TRUE OF. For example, consider the following:

```
MOUSE mx%, my%, button%
CASE TRUE OF
   WHEN FNin(100,100,200,200):PROCleft
   WHEN FNin(100,300,200,400):PROCright
   WHEN FNin(300,100,400,200):PROCdown
   WHEN FNin(300,300,400,400):PROCup
ENDCASE
```

This a reversal of the normal situation as the value (fixed) is given in CASE-OF, and the variables (or expressions) being tested follow the WHEN statements. FNin is assumed to be a function which tests whether (mx%,my%) lies within the rectangle specified (there is a similar example in in the PalDef256 program in last month's RISC User). In the example, the first occurrence of FNin which evaluates as *TRUE* is the path (and the only path) which will be followed. Interestingly, TRUE in the CASE statement may also be replaced by FALSE, or indeed by any fixed value (e.g. CASE 5 OF).

The priority implied by the ordering of WHEN statements can be very useful, and permits a more simplified coding of some examples than might otherwise be the case. For example:

```
CASE TRUE OF
   WHEN x>1000: PROCone
   WHEN x>750 : PROCtwo
   WHEN x>500 : PROCthree
ENDCASE
```

This effectively checks for x>1000, 750<x<=1000, and 500<x<=750, but in the latter two cases only one test is needed.

The CASE statement is a highly structured and useful addition to BBC Basic. It does also have its limitations, so a thorough understanding is essential if you are to exploit it to the full. Try it out when you can, but make sure you check the User Guide if you are still unsure of the correct syntax. RU

HINTS & TIPS HINTS & TIPS

Some more useful utilities selected by Lee Calcraft

305 SIMULATION

If you have a 310, and wish to see whether a particular program will run on a 305, then use:

```
    *CON.RAMFS.64
```

followed by Ctrl-Break. This will "waste" 512K of RAM by configuring it for RAM filing system use. Use *CON.RAMFS.0 to restore the status quo.

SMART COMPACTING

The command *COMPACT can be used to re-organise the free space on a disc so that it is grouped together. Unfortunately you may have to issue the command several times before compacting is complete. The following EXEC file will do the work for you, and will display the disc's space map, and free space on completion. It works by repeatedly calling *COMPACT until the time taken for the last call was less than 0.1 sec. The EXEC file itself should be given the name "COM". To help you to keep track of what is going on it also issues a beep at the start of each new attack.

```
*| >COM
*|        Auto Compact
*|=============================
*|   Compacting Please Wait
*|=============================
VDU21
SOUND 1,-15,150,3
VDU6:TIME=0:OSCLI("COMPACT"):VDU21
IF TIME>10 THEN *COM
VDU6,10:P."New Map:":VDU21
VDU6:OSCLI("MAP"):P.:OSCLI("FREE")
```

SCREEN PARAMETERS

The following function returns the start address of the Archimedes screen RAM. On a default configured 310 this is &1FD8000, but if more or less than 160K are allocated to the screen, or if the screen is scrolled at any time, the address returned will reflect this. The program which calls this function should include the following line prior to the call:

```
   DIM base 20
```

If you alter the number 148 in line 40 of the function definition to 7 or to 150, it will return the size of the currently selected screen mode, or the size of RAM currently allocated to screen use, respectively. The function makes use of the very useful "OS_ReadVduVariables" call. For further details, see the *Programmer's Reference Manual* part 1, page 122.

```
10REM >FnFindScr2
20:
30DEFFNfindscrn
40!base=148:base!4=-1
50SYS &31,base,base+8
60=base!8
```

EXEC FILE NOTES

Some of this month's hints involve EXEC files. An EXEC file is an ASCII file, without line numbers. It may be created using Twin, Wordwise or View, but not ArcWriter or Graphic Writer. Alternatively, issue the command ***BUILD filename**, and then type in the listing, hitting Escape after pressing Return on the last line. The numbers which appear on screen are not saved in the file. The main snag with *BUILD is that you cannot correct errors once you have pressed Return on a given line.

Once the file has been created, whether using *BUILD or whatever, you should set its file type to "Command" with:

```
   *SetType filename FFE
```

By saving the file to the LIBRARY directory of your disc you will ensure that the command is always available whenever you type:

```
   *Filename
```

SAVING AND LOADING CONFIGURATION DATA

If your machine ever gets "de-configured", or if you use different configurations at different times, this pair of utilities will prove invaluable. The two EXEC files below give you two new star commands: *CSAVE and *CLOAD. When called, each requests a filename. When supplied, the first will save all CMOS settings to the filename given. The second will load them back in again. There is no restriction on the number of configuration files you keep. But it is a good idea to keep the two EXEC files in the Library directory of your disc. The format of the files used is identical to that used by our CMOS Manager (RISC User Issue 2), and they are completely interchangeable.

HINTS & TIPS HINTS & TIPS

```
*| >CSave
*| Configuration Save ver 0.2
*|
VDU21
VDU6:INPUT "Save filename ",file$:D%=
  OPENOUT(file$):FORA=0TO239:SYS6,161
  ,A TO,,cmos:BPUT#D%,cmos:NEXT:CLOSE
  #D%:P."Configuration settings saved"

*| >CLoad
*| Configuration loader ver 0.3
*|
VDU21
VDU6:INPUT "Load filename ",file$:D%=
  OPENIN(file$):FORA=0TO239:OSCLI("FX
  162,"+STR$(A)+","+STR$(BGET#D%)):NE
  XT:CLOSE#D%:P."Now press Ctrl-Break"
```

RESTFUL COLOURS

For more restful and legible program listings, try the following function key definition:

```
    *KEY1 MODE12:COL.128,128,128,128:VDU19
,0,24,128,128,128|M
```

This will cause function key f1 to create a grey background with matching screen border. By varying any of the "128"s except for the first, you can adjust the colours to suit your own taste. If you use the Desktop, you will need to execute *FX225,1 each time you quit, in order to restore the function keys - as it says in RISC User Issue 3. Thanks to Greg Herdman for this hint.

ADFS MENU WHOLE-DIRECTORY COPYING

Copying whole directories complete with their nested contents is easily accomplished using the RISC User ADFS menu by marking the directory rather than its individual files. But remember, you should not supply the destination directory name as part of the pathname. So to copy the contents of $.LIBRARY into $.LIBRARY on a second disc, the destination should be given simply as :0.$ (or :1.$ as appropriate).

DIRECTORIES LAST - AGAIN

Barry Christie points out that last month's hint on using the rather odd "Shift-£" character as the first character in a directory name to force it to be listed last, sometimes causes problems from Basic since its ASCII value can be mistaken for a token. To avoid this, use the tilde character (~) instead (at the top left of the keyboard).

SOME USEFUL ALIASES

The following 12 aliases may prove useful. They set up 12 new star commands. The first 10 are for Epson-compatible printers, and create star commands whose name immediately follows the "ALIAS$" in any given line. The first is *INDENT. It will cause your printer to indent all output by 5 characters. The second (*CINDENT) cancels this. HASH and POUND switch between a hash and and English pound. BOLD, ELITE and PICA set up bold, elite and normal typefaces, while SKIP instigates automatic skipping of printout at the end of each page, avoiding printing across the perforations. CSKIP cancels this.

Finally *FXDUMP calls the resident Hardcopy module to produce a mono screendump of the current screen, and *CATALL will catalogue a complete disc giving the names and lengths of every file which it contains.

```
*| >AliasHints
*| version 0.2
*|
*| Printer commands
*SET ALIAS$INDENT ECHO |<2>|<1>|<27>|<1>|<108>|<1>|<5>|<3>
*SET ALIAS$CINDENT ECHO |<2>|<1>|<27>|<1>|<108>|<1>|<0>|<3>
*SET ALIAS$HASH ECHO |<2>|<1>|<27>|<1>|<82>|<1>|<0>|<3>
*SET ALIAS$POUND ECHO |<2>|<1>|<27>|<1>|<82>|<1>|<3>|<3>
*SET ALIAS$BOLD ECHO |<2>|<1>|<27>|<1>|<33>|<1>|<8>|<3>
*SET ALIAS$ELITE ECHO |<2>|<1>|<27>|<1>|<33>|<1>|<19>|<3>
*SET ALIAS$PICA ECHO |<2>|<1>|<27>|<1>|<33>|<1>|<0>|<3>
*SET ALIAS$SKIP ECHO |<2>|<1>|<27>|<1>|<78>|<1>|<6>|<3>
*SET ALIAS$CSKIP ECHO |<2>|<1>|<27>|<1>|<79>|<3>
*| Screen dump
*SET ALIAS$FXDUMP HARDCOPYFX 1 1 1 0 5
*| CatAll
*SET ALIAS$CATALL COUNT $ V
```

RISC USER magazine

MEMBERSHIP

RISC User is available only on subscription at the rates shown below. Full subscribers to RISC User may also take out a reduced rate subscription to BEEBUG (the magazine for the BBC micro and Master series).

All subscriptions, including overseas, should be in pounds sterling. We will also accept payment by Connect, Access and Visa, and official UK orders are welcome.

RISC USER SUBSCRIPTION RATES		RISC USER & BEEBUG
£14.50	1 year (10 issues) UK, BFPO, Ch.I	£23.00
£20.00	Rest of Europe & Eire	£33.00
£25.00	Middle East	£40.00
£27.00	Americas & Africa	£44.00
£29.00	Elsewhere	£48.00

BACK ISSUES

We intend to maintain stocks of back issues New subscribers can therefore obtain earlier copies to provide a complete set from Vol.1 Issue 1. Back issues cost £1.20 each. You should also include postage as shown:

Destination	UK, BFPO, Ch.Is	Europe plus Eire	Elsewhere
First Issue	40p	75p	£2
Each subsequent Issue	20p	45p	85p

MAGAZINE DISC

The programs from each issue of RISC User are available on a monthly 3.5" disc. This will be available to order, or you may take out a subscription to ensure that the disc arrives at the same time as the magazine. The first issue (with six programs and animated graphics demo) is at the special low price of £3.75. The disc for each issue contains all the programs from the magazine, together with a number of additional items by way of demonstration, all at the standard rate of £4.75.

MAGAZINE DISC PRICES	UK	Overseas
Single Issue discs	£ 4.75	£ 4.75
Six months subscription	£25.50	£30.00
Twelve months subscription	£50.00	£56.00

Disc subscriptions include postage, but you should add 50p per disc for individual orders.

All orders, subscriptions and other correspondence should be addressed to:

RISC User, Dolphin Place, Holywell Hill, St Albans, Herts AL1 1EX.
Telephone: St Albans (0727) 40303
(24hrs answerphone service for payment by Connect, Access or Visa card)